**OBOR**

**18-20-M/01 INFORMAČNÍ TECHNOLOGIE**

**(ŠVP: Informační technologie)**

**MATURITNÍ PROJEKT**

TÉMA PRÁCE

**TÉMA PROJEKTU (ČERVENÝ TEXT NAHRAĎTĚ)**

2020 AUTOR (PŘÍJMENÍ Jméno)

**MÍSTO TOHOTO LISTU VLOŽTE LIST SE ZADÁNÍM**

**PROHLÁŠENÍ**

Prohlašuji, že jsem svou maturitní práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW, atd.) uvedené v seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 10. 4. 2020 ……………………………………

podpis

**PODĚKOVÁNÍ**

Text poděkování …. Běžně se děkuje rodinným příslušníkům za jejich morální podporu, vedoucí práce za její vedení, dalším konkrétním učitelům a konzultantům za věcné připomínky atd.

**ANOTACE**

|  |  |
| --- | --- |
| Jméno autora |  |
| Název maturitní práce | Musí se shodovat s oficiálním názvem práce v zadání |
| Rozsah práce | Počet stran práce ve formátu (např. 165 s. + 50 s. příloh) |
| Školní rok vyhotovení | 2020 |
| Škola | Střední průmyslová škola na Proseku |
| Vedoucí práce |  |
| Využití práce | Zde popište použití práce, např. výukové účely v oboru Telekomunikace v předmětu tom a tom |
| Abstrakt | Např. Práce je zaměřena na návrh webové aplikace pro evidenci … |
| Klíčová slova | Zde uveďte klíčová slova práce, tj. slova, která definují její obsah, např. webové stránky, skriptovací jazyky, syntaxe,… |

**ANNOTATION**

|  |  |
| --- | --- |
| Autor |  |
| Title of graduation work |  |
| Extend | Např. 165 p. + 50 p. of appendices |
| Academic year | 2020 |
| School | Střední průmyslová škola na Proseku |
| Supervisor |  |
| Application |  |
| Abstrakt |  |
| Key words |  |
|  |  |

**Obsah**

[ÚVOD 8](#_Toc18496332)

[1 KAPITOLA (NADPIS 1. KAPITOLY) 8](#_Toc18496333)

[1.1 NADPIS 1. PODKAPITOLY 8](#_Toc18496334)

[1.1.1 NADPIS PODKAPITOLY 8](#_Toc18496335)

[2 Další kapitoly a podkapitoly 8](#_Toc18496336)

[ZÁVĚR 10](#_Toc18496337)

[POUŽITÁ LITERATURA 11](#_Toc18496338)

[SEZNAM OBRÁZKŮ 12](#_Toc18496339)

ÚVOD

Text

# KAPITOLA (NADPIS 1. KAPITOLY)

Název kapitol bez teček a každá na nové stránce.

Text (obr. 1) – „obr. 1“ vloženo jako křížový odkaz ze záložky Vložení, vkládá se pouze návěstí a číslo od každého obrázku

obr. – popis obrázku (vloženo jako titulek ze záložky reference, musí být pod každým obrázkem)

## NADPIS 1. PODKAPITOLY

Text

### NADPIS PODKAPITOLY

Text

# Další kapitoly a podkapitoly

Úvod

Jako téma své maturitní práce jsem si zvolil herní aplikaci s tématem ‚‚Roguelike hra s procedurální generací‘‘. Cílem mé práce bude vytvoření náhodné generace prostředí, nepřátel a předmětů, k čemuž použiji převážně algoritmy procedurální generace.

Na trhu je podobných herních aplikací v moderní době již více, ale jen malé části z nich se daří zcela naplňovat tyto cíle. Některé z nich mají například strukturu prostředí předem danou, a náhodně generují pouze její části, či mají tak striktně omezené spektrum získatelných předmětů či nepřátel, na které můžete narazit, dle úrovně, že na tyto předměty či nepřátele v určitých částech hry jednoduše nemáte šanci narazit. Oproti těmto aplikacím bych chtěl mít všechny herní mechaniky, které mají být založené na náhodě, opravdu zcela náhodné. To mi pomůže zaručit, že každý nový průchod hrou bude jiný.

Generaci prostředí každé úrovně plánuji provádět pouze na základě určitých parametrů, které navíc nebudou přímo dané, například místo přesné velikosti místností bude určena pouze maximální a minimální velikost místností. Zároveň ale budu řešit, jak docílit toho, aby mezi prostředími nebyly nevyvážené rozdíly. Dále bych se chtěl zaměřit na to, aby má aplikace byla ve výsledku komplexní, ať už graficky, hratelností či zvukově. Za velký úspěch bych totiž považoval i výsledné vydání hry na platformu Steam.

Aplikaci budu vyvíjet v prostředí Unity, což je známý herní engine, hojně používaný velkými i menšími herními studii.

# Unity

## Úvod

Unity je multiplatformní herní engine, za kterým stojí společnost Unity Technologies, a používá se především k vývoji herních aplikací a simulací pro počítače, konzole i mobilní zařízení. Unity je kombinací herního engine, který umožňuje spuštění vytvořené herní aplikace, a IDE, neboli vývojového prostředí, jenž umožňuje grafický náhled.

## Kód

Při instalaci Unity navíc uživatel dostane možnost nainstalovat ještě poslední klíčový nástroj k vývoji herních aplikací - editor kódu, v tomto případě Microsoft Visual Studio. Editor kódu ale může být nahrazen dalšími oficiálně podporovanými editory, jako je Visual Studio Code či JetBrains Rider, avšak uživatel může použít i jakýkoli jiný oficiálně nepodporovaný editor i bez vlastního kompilátoru kódu, protože Unity má svůj vlastní kompilátor. Unity nabízí možnost vývoje herních aplikací v programovacím jazyce C# pro počítače, herní konzole, mobilní telefony, virtuální realitu, rozšířenou realitu i webové platformy.

## Cena

Jednou z hlavních výhod Unity je cena tohoto programu. Standardní plán je totiž zdarma, dokud uživatel splňuje podmínku, že jeho příjem z komerčního využití je nižší než 100 000 amerických dolarů za posledních 12 měsíců. Toto je pro mnoho, zvláště menších vývojářů, velmi lákavá nabídka.

## Platformy

Dalším přínosem pro uživatele Unity jsou jeho multiplatformní vlastnosti umožňující spuštění herní aplikace vyvinuté v Unity jak na počítači s operačním systémem Windows, tak například na mobilním telefonu či zařízení s macOS.

## Unity Asset Store

Nelze také opomenout Unity Asset Store, což je webová stránka s knihovnou tisíců placených i neplacených prostředků, které šetří čas a námahu. Tyto prostředky vytváří společnost Unity Technologies i členové komunity a publikují je právě v Unity Asset Store. Jsou zde k dispozici různé typy prostředků, od textur, animací, modelů, skriptů obsahujících hotová řešení určitých problémů, výukových programů až po celé příklady projektů. Kromě těchto typů prostředků existují dokonce i prostředky rozšiřující nabídky a komponenty samotného editoru o další položky a panely odpovídající novým nebo rozšířeným funkcím, jenž usnadňují specifické potřeby či umožňují něco, co editor sám o sobě neumožňuje. Unity Asset Store také uživatelům umožňuje stát se vydavatelem a prodávat či poskytovat své výtvory ostatním členům komunity.

## Animace

A v poslední řadě je Unity populární dokonce i mezi umělci díky možnostem vytváření 3D a 2D animací od základu. Unity totiž disponuje výkonnými animačními nástroji a zároveň téměř vše v Unity je animovatelné.

## Dokumentace

Co se dokumentace týče, tak Unity má poměrně rozsáhlou a přehlednou dokumentaci s příklady a ukázkami. Nechybí ani výukové programy přímo od společnosti Unity Technologies a rozsáhlé sady knih a výukových programů poskytovaných třetími stranami.

## Důležité pojmy v Unity

### ScriptableObject

ScriptableObject je datový kontejner, který lze použít k ukládání velkého množství dat nezávisle na počtu instancí třídy. Jednou z hlavních výhod použití ScriptableObject je snížení spotřeby paměti tím, že se vyhnete kopírování hodnot. To je zvlášť užitečné, pokud se používá velké množství Prefab, jenž ukládají neměnná data v připojených skriptech. Pokaždé, když je z Prefab vytvořena nová instance, získá svou vlastní kopii těchto dat. Narozdíl od toho ScriptableObject umožňuje uložení dat a následné přistupování k těmto datům pomocí odkazu ze všech instancí daného Prefab. To znamená, že v paměti je pouze jedna kopie dat.

### GameObject

GameObject je jedním z nejdůležitějších pojmů v editoru Unity. Každý objekt v Unity je GameObject, jak postavy, předměty a kamery, tak i světla, efekty a další. Avšak i navzdory tomu, že GameObject jsou základní objekty v Unity, představující téměř vše, samy o sobě moc funkcionality nemají. Fungují totiž jako kontejnery pro komponenty, které požadovanou funkcionalitu implementují.

### Abstract

Abstraktní modifikátor označuje chybějící nebo neúplnou implementaci. Modifikátor Abstract lze použít u tříd, metod, vlastností, indexerů i událostí. Používá se pro deklaraci třídy, jenž je určena pouze jako základní třída pro další třídy, nikoliv jako samostatná instance. Členy označené jako Abstract je nutné implementovat jinými než Abstract třídami, které jsou z dané Abstract třídy odvozeny.

### Awake

Metoda Awake je volána pouze jednou, a to při načítání instance skriptu. Volá se buď při inicializaci aktivního GameObject, který obsahuje skript, při načítání scény, když je dříve neaktivní GameObject nastaven na aktivní, nebo po inicializaci GameObject. Metoda Awake se používá k inicializaci proměnných či stavů před spuštěním aplikace.

### Start

Metoda Start se volá hned v prvním snímku, těsně před prvním voláním jakékoli z metod z metody Update. Stejně jako metoda Awake je metoda Start zavolána přesně jednou. Metoda Awake je ale volána při inicializaci objektu skriptu bez ohledu na to, zda je skript povolen, či nikoli. Oproti tomu metoda Start není volána pokud skript není v době inicializace povolen.

### Update

Metoda Update je volána každý snímek. Update je sice nejčastěji používaná metoda jakéhokoli herního skriptu, nicméně ne každý herní skript potřebuje metodu Update.

### FixedUpdate

Metoda FixedUpdate je podobná metodě Update, ovšem s jedním důležitým rozdílem, má totiž frekvenci fyzikálního systému. To znamená, že je pevně nastavená snímková frekvence, ve které je metoda FixedUpdate volána, typicky 50 volání za sekundu. Tato metoda se používá hlavně pro přesné fyzické kalkulace.

### Transform

Transform je poloha, rotace a měřítko objektu. Každý objekt ve scéně má vlastní Transform. Slouží k ukládání a k manipulaci s polohou, rotací a měřítkem objektu. Každý Transform může mít nadřazený objekt, což umožňuje tyto vlastnosti aplikovat hierarchicky.

### Rigidbody

Přidáním komponenty Rigidbody k objektu začne být jeho pohyb řízen fyzikálním enginem Unity. I bez přidání jakéhokoli kódu bude objekt s komponentou Rigidbody tažen gravitací směrem dolů, a pokud se správně nastaví další komponenty, bude reagovat například na kolize s dalšími objekty.

### Sprite

Sprite reprezentuje textury 2D objektů. Používají se pro grafiku postav, předmětů, projektilů a dalších prvků 2D her.

### Component

Základní třída pro vše, co se připojuje ke GameObject.

zdroje:

https://conceptartempire.com/what-is-unity/

https://designagame.eu/2013/12/unity-popular-videogame-development/

https://unity3d.com/quick-guide-to-unity-asset-store

https://docs.unity3d.com/Manual/class-ScriptableObject.html

https://docs.unity3d.com/Manual/GameObjects.html

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/abstract

https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html

https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html

https://docs.unity3d.com/ScriptReference/Transform.html

https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html

https://docs.unity3d.com/ScriptReference/Rigidbody.html

https://docs.unity3d.com/ScriptReference/Sprite.html

https://docs.unity3d.com/ScriptReference/Component.html

https://docs.unity3d.com/Manual/index.html

# Adobe Photoshop

## Úvod

Adobe Photoshop je jedním z nejpoužívanějších programů pro úpravu fotografií, rastrových obrázků, ilustrací, retušování fotografií, tvorbu digitálního umění a úpravu mnoha formátů obrazových i video souborů, vytvořený společností Adobe Inc. Oficiální verzi tohoto programu lze používat jak na počítačích se systémem Windows tak i MacOS. Pomocí programu Adobe Photoshop lze například měnit pozadí obrázku či simulovat skutečný obrázek. Nástroje v tomto programu umožňují upravovat jak jednotlivé snímky, tak i velké množství fotografií.

## Vrstvy

Adobe Photoshop je speciálně navržen a vyvinut tak, aby uživatelům umožňoval vytváření a úpravu rastrových obrázků ve více vrstvách. Tyto vrstvy podporují například průhlednost či mohou fungovat jako masky a filtry, které mění podkladové obrázky ve vrstvách pod nimi, čímž zajišťují zachování podkladového obrázku. Na vrstvy lze aplikovat stíny a další efekty, jako je alfa kompozice a několik barevných modelů (CMYK, RGB, přímé barvy a barevný prostor Duotone a Lap). Pro zachování všech funkcí včetně efektů, masek a vrstev, se standardně používá formát souboru .PSD (Photoshop Document), pro velké soubory pak formát souboru .PSB (Photoshop Big).

## Uživatelé

Adobe Photoshop je na profesionální úrovni využíván hlavně designéry, webovými vývojáři, grafiky a fotografy. Digitální nebo naskenované obrázky lze upravovat pro použití online nebo v tisku. Dále lze v tomto programu vytvářet rozvržení webových stránek pro vytvoření detailních návrhů předtím, než vývojáři přejdou do fáze samostatného kódování. Jednotlivé grafické prvky lze vytvářet a exportovat i pro použití v jiných programech.

## Verze

Existuje několik verzí programu Adobe Photoshop, jako je Photoshop CC, Photoshop Elements, Photoshop Lightroom a Photoshop Express. Lze použít i oficiální verzi programu Photoshop pro iOS, avšak ta má oproti ostatním verzím omezené funkce. Adobe Photoshop je k dispozici samostatně jako předplatné, které zahrnuje i Photoshop Lightroom, nebo jako součást většího předplatného Creative Cloud.

## Výuka

Existuje mnoho možností, jak se naučit používat Adobe Photoshop. Od osobních kurzů Photoshopu, živých online kurzů Photoshopu až po výuku prostřednictvím online výukových programů Photoshopu či knih o Photoshopu.

## Předplatné

Předplatné Adobe Photoshop je možné zakoupit za 12,09 €/měsíc (144,33 €/rok) s 20 GB místa v cloud úložišti nebo za 24,19 €/měsíc (288,80 €/rok) s 1TB místa v cloud úložišti. Ceny obou předplatných jsou včetně DPH, k oběma verzím předplatného kupující dále obdrží předplatné programů Lightroom a Lightroom Classic a jsou na rok. K dispozici je také již zmiňované předplatné všech aplikací Creative Cloud, které obsahuje Adobe Photoshop spolu se všemi ostatními aplikacemi od Adobe Inc., a lze zakoupit za 60,49 €/měsíc (725,85 €/rok) včetně DPH. Zvýhodněné plány předplatných jsou k dispozici pouze pro studenty, učitele či organizace, které chtějí zakoupit licence pro skupiny 10 a více uživatelů. Všechny ceny jsou platné ke dni 13.12.2021.

## Porovnání s programem GIMP

I přestože Adobe Photoshop je průmyslový standard pro úpravu fotografií, rastrových obrázků, digitálního umění a mnoha dalších, není to jediný program na trhu. V dnešní době je na trhu velké množství jiných podobných programů, jenž se buď specializují pouze na určité potřeby uživatelů, nebo se snaží být v tomto ohledu stejně univerzální jako Adobe Photoshop. Primárními faktory při rozhodování, jaký z těchto programů vybrat, jsou cena a kompatibilita. V těchto ohledech se stává největším soupeřem Adobe Photoshop program GIMP.

Program GIMP totiž v pár oblastech zcela vítězí nad Adobe Photoshop. Je kompletně zdarma a open-source, což znamená, že kdokoli si může prohlédnout jeho zdrojový kód a dokonce vyvinout vlastní vylepšení či rozšíření. Druhou podstatnou výhodou je, že program GIMP narozdíl od Adobe Photoshop funguje i na operačním systému Linux. Má také samozřejmě rozsáhlou sadu základních nástrojů pro úpravu obrázků.

Prakticky ve všech ostatních ohledech již ale vítězí Adobe Photoshop. Co se týká nástrojů pro výběr, Adobe Photoshop vyhrává díky působivým nástrojům pro automatický výběr, které výrazně urychlí pracovní postup. U záplat a klonování jsou oba sice poměrně vyrovnané, avšak u automatických úprav znovu vítězí Adobe Photoshop, neboť program GIMP má problémy i s automatickou korekcí jasu a kontrastu. S běžnými formáty obrázků nemá potíže ani jeden program, ale při úpravě RAW fotografií zase vítězí Adobe Photoshop, tentokrát díky zefektivněnému procesu pro zpracování RAW fotografií a pravidelným aktualizacím, s nimiž přichází podpora nejnovějších fotoaparátů a typů RAW. Stejně tomu je i u uživatelského rozhraní, kde opět vítězí Adobe Photoshop, a to díky velmi užitečné možnosti nastavení více vlastních přednastavených pracovních prostorů pro různé typy úprav obrázků.

Odpovědí na otázku, který program zvolit, tedy je, že Adobe Photoshop je o velký krok napřed před programem GIMP. Má výkonnější funkce, lepší podporu a také obrovskou komunitu uživatelů. Pokud je potřeba editor obrázků na profesionální úrovni, je Adobe Photoshop jednoznačným vítězem. Adobe Photoshop má navíc několikadenní zkušební verzi, která je zcela zdarma.

zdroje:

https://www.adobe.com/cz/creativecloud/plans.html?plan=individual&filter=all

https://www.adobe.com/cz/products/photoshop.html

https://www.agitraining.com/design-news/photoshop-training-news/what-photoshop

https://www.techopedia.com/definition/32364/adobe-photoshop

https://thegimptutorials.com/gimp-vs-photoshop/

https://www.designbuckle.com/adobe-photoshop-vs-gimp/

Praktická část

# Průzkum trhu

Průzkum trhu s podobnými herními aplikacemi, přesněji roguelike videohrami, jsem prováděl hlavně na platformě Steam a Epic Games Launcher. Již delší dobu se o podobné herní aplikace zajímám, a sám jsem jich několik vyzkoušel. Roguelike je jeden z nejstarších žánrů videoher a lze považovat za podžánr žánru RPG.

Do hlavní charakteristiky moderních roguelike videoher patří například stylizovaná grafika, procedurální generace prostředí či úrovní, permanentní smrt hráčovy postavy, identifikace neznámých předmětů nebo i singleplayer.

Procedurální generace prostředí zajišťuje, že každý průchod videohrou bude lehce nebo zcela jiný než ten minulý, což má velký vliv na takzvané “replay value”, neboli zachování hodnoty videohry při opakovaném hraní či dokončení.

Permanentní smrt hráčovy postavy nutí hráče k velké obezřetnosti, poučování se z předchozích chyb a celkově to zesiluje atmosféru videohry.

Identifikace neznámých předmětů je založená na reálném životě, když člověk najde nějaký neznámý předmět, dokáže většinou odhadnout, o co se přibližně jedná, ale nezná všechny jeho vlastnosti či funkce. Ve videohrách funguje identifikace neznámých předmětů obdobně, hráč se typicky po nalezení předmětu hned nemusí dozvědět, o co se přesně jedná, ale například pouze to, že nalezený předmět je meč. Až po identifikaci nalezeného předmětu se hráč dozví o skrytých vlastnostech předmětu, čímž může být například schopnost střílení blesků na nepřátele, ale také prokletí, což může hráčovi velmi znepříjemnit průchod hrou.

Jako jednu z hlavních charakteristik moderních roguelike videoher jsem uvedl i singleplayer, neboli mód pouze pro jednoho hráče. S tímto jsem se setkal u většiny videoher, jenž jsem vyzkoušel, a u některých mě to docela mrzelo. Důvodem je pravděpodobně přílišná náročnost zprovoznění všech herních mechanik do módu pro více hráčů, což je u malých videoherních vývojářů, kteří většinou za roguelike videohrami stojí, pochopitelné.

Díky svému průzkumu trhu jsem se rozhodl například pro implementaci procedurální generace prostředí, stylizovanou grafikou, permanentní smrt hráčovy postavy ale také pro zavržení implementace identifikace neznámých předmětů, neboť mě tato mechanika ve vyzkoušených videohrách spíše otravovala.

Jak již popisuji v úvodu své práce, jako díru na trhu roguelike videoher vidím nedostatečnou náhodnost. Ve své herní aplikaci se proto snažím docílit nepředvídatelného zážitku z každého průchodu hrou pro co největší replay value. Procedurální generace je klíčem k řešení problému náhodnosti každé úrovně, místnosti i objektů uvnitř. Avšak ani tyto parametry procedurální generace nenechávám pevně nastavené, naopak zde dávám možnost projevit se dalším náhodným prvkům. Každá úroveň je opravdu náhodná jak v počtu místností, tak například ve velikostech místností, a to zajišťuje jedinečnost jednotlivých úrovní zároveň s požadavky na zvolení jiné strategie pro úspěšný průchod.

# Programovací jazyk

Pro tvorbu herní aplikace jsem zvolil vysokoúrovňový objektově orientovaný programovací jazyk C#. Za tímto programovacím jazykem stojí společnost Microsoft. C# běží na platformě .NET Framework, na jehož vývoj se již zmíněná společnost Microsoft spojila ještě se společností Xamarin.

C# je typově bezpečný jazyk, což znamená, že neumožňuje typové konverze, které by mohly vést ke ztrátě dat nebo jiným problémům. Tento programovací jazyk jsem si vybral neboť s ním mám nejvíce zkušeností a většinu jeho hlavních vlastností považuji za výhody.

zdroje:

https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

https://www.c-sharpcorner.com/article/what-is-c-sharp/

# Platforma

Jako cílovou platformu pro tuto herní aplikaci jsem vybral operační systém Microsoft Windows, zejména kvůli jeho rozšířenosti, neboť se jedná o jeden z nejrozšířenějších operačních systémů na světě. Zároveň má převládající postavení na trhu, což je opravdu nezanedbatelná výhoda pro vývojáře ať už herních tak i ostatních aplikací.

Z multiplatformního hlediska je tato herní aplikace určena pouze pro stolní počítače s operačním systémem Microsoft Windows. Hlavním důvodem je potřebný výkon ke správnému běhu aplikace, který je ovlivněn hlavně použitou procedurální generací, dále pak grafickými nastaveními a vysokému počtu objektů, kde u většiny z nich je potřeba počítat fyzické simulace. I kdybych problém s potřebným výkonem vyřešil, zůstaly by zde potíže s předěláním ovládání například pro mobilní telefony. Pro správné používání této herní aplikace je totiž nezbytná možnost rychlého a přesného ovládání pomocí několika kláves a myši, což je na obvyklých mobilních telefonech takřka nemožné.

# Použité nástroje

## Microsoft Visual Studio Community 2019

Editor kódu od společnosti Microsoft. Microsoft Visual Studio bylo využito pro psaní a editaci kódu během celého vývoje herní aplikace.

## Unity

Program, jenž je kombinací herního engine a vývojového prostředí. Unity bylo použito jako herní engine herní aplikace, vývojové prostředí herní aplikace a jako kompilátor kódu.

## Adobe Photoshop CC 2020

Program pro tvorbu a úpravy bitmapové grafiky, fotografií, rastrových obrázků a mnoho dalšího. Adobe Photoshop byl použit pro tvorbu grafiky herní aplikace, a to převážně pro tvorbu textur.

## Procedurální generace

### Algoritmy

Při tvorbě prostředí jsem se rozhodoval mezi algoritmy Random Walk a Binary Space Partitioning.

### Random Walk

Random Walk je poměrně jednoduchý algoritmus, při kterém se souřadnice startovní pozice postupně rozšiřují do náhodných směrů.

### Binary Space Partitioning

Binary Space Partitioning je o něco složitější algoritmus. Jde o postupné rozdělování prostoru na menší a menší “podprostory”.

Poté se mezi jednotlivými podprostory vytvoří místo a mohou být například spojeny chodbami.

Nakonec jsem se rozhodl pro použití algoritmu Random Walk, neboť mi více vyhovoval výsledný vzhled jednotlivých místností. Dalším možným řešením by byla i kombinace obou algoritmů, avšak to mi přišlo zbytečné, neboť jsem se dostal k podobným, ne-li stejným výsledkům jen pomocí algoritmu Random Walk. Algoritmus Random Walk jsem dále využil i pro tvorbu koridorů (chodeb), které propojují jednotlivé místnosti.

zdroj:

https://medium.com/@guribemontero/dungeon-generation-using-binary-space-trees-47d4a668e2d0

https://www.jamestease.co.uk/blether/procedural-generation-with-binary-space-partitions-and-rust/

https://www.mathworks.com/matlabcentral/fileexchange/51260-random-walk-using-monte-carlo-randomized-algorithm

### Třída ProceduralGenerationAlgorithms

V této třídě jsou definovány metody obsahující algoritmy, na kterých stojí implementovaná procedurální generace.

### Metoda RandomWalk

Obsahuje základní algoritmus pro většinu procedurální generace obsažené v této herní aplikaci. Metoda RandomWalk vyžaduje parametr se startovní pozicí a “délkou chůze”. Hlavní práci zde odvádí jednoduchá smyčka, která souřadnice startovní pozice vždy rozšíří do náhodného směru a uloží do kolekce typu HashSet. Poté to samé provede s dalšími pozicemi, které se vždy odvíjejí od poslední uložené pozice. Počet pozic, který tato smyčka vygeneruje je založený na parametru s “délkou chůze”.

### Metoda RandomWalkCorridor

Tato metoda je upravená verze metody RandomWalk. Metoda RandomWalkCorridor vyžaduje parametr se startovní pozicí, délkou generovaného koridoru a směrem. Stejně jako v metodě RandomWalk je zde smyčka, která souřadnice startovní pozice vždy rozšíří do směru daného parametrem se směrem a uloží je do kolekce typu List. Počet pozic, který tato smyčka vygeneruje je založený na parametru s délkou generovaného koridoru. Pro ukládání pozic zde používám kolekci typu List neboť kolekce typu List si ukládá položky popořadě. V tomto případě je nutné ukládat položky kolekce popořadě, protože pro generaci koridorů je potřeba znát první (začátek) a poslední (konec) pozici tvořící daný koridor.

## Inventář a předměty

### Třída Inventory

V této třídě jsou definovány metody SwitchSlot a TryAddItem, jenž obstarávají základní funkcionalitu všech inventářů. Vlastní inventář má hráč, ale i například truhla s předměty.

### Metoda SwitchSlot

Základní metoda pro přesouvání předmětů mezi jednotlivými inventáři. Vyžaduje dva parametry, parametr určující do jakého slotu inventáře se bude předmět přesouvat a parametr určující jaký předmět se bude přesouvat. Tato metoda dále vrací proměnnou obsahující předmět, jenž byl původně na slotu před tím, než byl nahrazen.

### Metoda TryAddItem

Vyžaduje parametr obsahující předmět. Zjišťuje, zda lze určený předmět přidat do inventáře, pro který byla zavolána. Prochází postupně sloty daného inventáře, a pokud je některý volný, vloží do něj určený předmět.

### Konstruktor Inventory

Vyžaduje parametr s velikostí inventáře, který později určuje počet slotů v daném inventáři.

### Třída InventoryVisualizer

Zařizuje vizualizaci a dodatečnou funkcionalitu inventářů. V metodě Awake je obsažena konfigurace hráčova inventáře, přiřazení startovních předmětů a načítání Sprite předmětů. Je zde definována metoda InventoryChanged, která se používá pro aktualizaci a načtení předmětů v inventáři po jakékoliv manipulaci s předměty mezi inventáři. Metody OpenInventory a CloseInventory slouží pro vizualizaci otevírání a zavírání inventáře. V metodě CloseInventory je použita metoda TryAddItem pro řešení situace, kdy hráč zmáčkne tlačítko pro zavření inventáře, ale zároveň drží kurzorem nějaký z předmětů, jenž se v tu chvíli nenachází ani v jednom inventáři. V tuto chvíli se metoda TryAddItem pokusí vložit držený předmět do inventáře hráče, a poté zpět do inventáře truhly, díky čemuž se předejde smazání drženého předmětu. Další klíčovou metodou je SwitchHoldingItemWithSlot, jenž umožňuje manipulaci s předměty mezi inventáři pomocí kurzoru myši. Metoda rozšiřující funkcionalitu inventářů se nazývá ShiftClick. Tato metoda umožňuje pohodlnější a rychlejší přesouvání předmětů mezi dvěma inventáři. Pokud hráč zmáčkne a drží klávesu Shift, může jednoduše klikat myší na předměty v inventáři, které se pak automaticky přesunou do druhého inventáře.

Třída Item

Základní třída pro veškeré předměty. Vzhledem k velkému počtu instancí třídy Item v herní aplikaci je tato třída zároveň ScriptableObject. Třída item má tři vlastnosti, první z nich je displayName, neboli název předmětu, který je viditelný pro hráče přímo v herní aplikaci. Dále je zde systemName, což je název předmětu, se kterým pracuje kód, bývá proto jednodušší. Poslední vlastností je řetězec lore, jenž má hlavně estetický význam, neboť obsahuje krátký příběh o daném předmětu.

Třída Wand

Třída pro základní a hlavní předmět postavy hráče, kouzelnou hůlku. Dědí ze třídy Item. Třída Wand má čtyři vlastnosti.

Vlastnost projectile

Představuje vystřelený projektil a je GameObject.

Vlastnost projectileSpeed

Rychlost vystřeleného projektilu.

Vlastnost projectilesPerSecond

Počet projektilů vystřelených za sekundu.

Vlastnost spread

Reprezentuje rozptyl vystřelených projektilů.

Třída Interactable

Základní třída pro všechny objekty v herní aplikaci, se kterými může hráč interagovat. Tato třída je abstract a má tři abstract metody.

Metoda Activate

Používá se pro zobrazení textu jenž informuje hráče o možnosti interakce.

Metoda Interact

Používá se pro spuštění interakce s objektem, například pro otevření okna zobrazující inventář truhly s předměty.

Metoda Deactivate

Používá se pro ukončení interakce s objektem, například pro zavření okna zobrazující inventář truhly s předměty.

Třída ChestInteractable

Třída obstarávající truhly s předměty, dědí ze třídy Interactable. Metody Activate, Interact a Deactivate jsou zde společně s proměnnou typu Inventory použity pro veškerou funkcionalitu.

Nepřátelé a umělá inteligence

Pathfinding

Pathfinding je hlavně ve videohrách termín, označující hledání cesty z libovolné souřadnice v herním světě na jinou. Pro vypočítání té nejoptimálnější cesty existuje mnoho různých algoritmů. Já jsem po krátkém prostudování několika z nich zvolil známý algoritmus nesoucí název A\*.

A\* algoritmus

A\* algoritmus je inteligentní algoritmus, což jej odlišuje od ostatních běžných algoritmů. Tato skutečnost je potvrzena i tím, že mnoho her a webových map využívá tento algoritmus k velmi efektivnímu nalezení nejkratší cesty.

Dobrým příkladem je čtvercová mřížka s několika překážkami, kde je jeden počáteční čtverec a jeden cílový čtverec. Úkolem je dostat se ze startovního čtverce co nejrychleji na čtverec cílový, pokud je to samozřejmě možné. A\* algoritmus funguje tak, že v každém kroku vybere čtverec podle hodnoty „f“, což je parametr roven součtu dvou dalších parametrů - „g“ a „h“. V každém kroku vybere algoritmus čtverec s nejnižší hodnotou „f“ a zpracuje ho. „g“ jsou náklady vynaložené na pohyb z výchozího čtverce do daného čtverce na mřížce po cestě, která byla vygenerována. „h“ jsou odhadované náklady na pohyb z daného čtverce na mřížce na konečný čtverec. Hodnota „h“ je v podstatě jakýsi odhad. Skutečná vzdálenost je totiž neznámá, dokud není nalezena celá cesta, protože v cestě mohou být nejrůznější překážky. Existuje mnoho způsobů, jak toto „h“ dále vypočítat.

Po implementaci tohoto algoritmu se objevil problém s tím, že se nepřátelé občasně zasekávali o zdi kvůli svému Collider. Tento problém jsem vyřešil odstraněním Collider z nepřátel, což jim sice umožnilo potenciálně procházet zdmi, avšak díky funkčnímu Pathfinding se

Třída Movement

Abstract třída pro pohyb nepřátel. Obsahuje definici abstract metody Move a konstruktor Movement.

Metoda Move

Vyžaduje parametr moveTo, neboli cíl, a parametr speed, jenž určuje rychlost pohybu.

Konstruktor Movement

Vyžaduje parametr transform, obsahující transform konkrétního nepřítele, a parametr rigidBody, určující RigidBody daného nepřítele.

Třída StraightMovement

Dědí ze třídy Movement. Je zde definována proměnná Movement vyžadující parametr transform. Dále je zde definována abstract metoda Move.

Třída PathMovement

Třída MeleeCombat

Třída Enemy

Dědí ze třídy Living. Třída Enemy slouží pro základní nastavení nepřátel. Obsahuje proměnnou movement pro pohyb, proměnnou combat pro boj, proměnnou speed pro rychlost pohybu a proměnnou rigidBody pro RigidBody nepřítele. V metodě Start se do proměnné rigidBody přiřadí RigidBody konkrétního nepřítele a do proměnné movement se přiřadí objekt PathMovement s nastavenými hodnotami. Dále se do proměnné combat přiřadí objekt MeleeCombat s nastavenými hodnotami a jako cíl se nastaví hráč. V metodě Update se při boji řeší pohyb nepřítele směrem k hráči prostřednictvím metody Move.

Pathfinding A\* algoritmus - popsat

Uživatelské rozhraní

Popsat script na stíny co jsme použili

ZÁVĚR

POUŽITÉ ZDROJE

Stránky pro generování citací dle normy:

Generátor citací na [www.citace.com](http://www.citace.com)

Seznam citací dle ISO 690 (např. dle [www.boldis.cz/citace/citace2.pdf](http://www.boldis.cz/citace/citace2.pdf))



SEZNAM OBRÁZKŮ

Seznam obrázků včetně zdrojů