Střední průmyslová škola Třebíč

Maturitní práce

WEBOVÁ APLIKACE PRO TVORBU 3D MAP

Profilová část maturitní zkoušky

Studijní obor: Informační technologie

Třída: ITA4

Školní rok: 2024/2025 Viktor Čada

Zadání práce

Střední průmyslová škola Třebíč
Manželů Curieových 734, 674 01 Třebíč

Zadání ročníkové práce

Obor studia: **18-20-M/01 Informační technologie**

Celé jméno studenta:	**Viktor Čada**		
Třída:	**ITA4**	Školní rok:	**2024/2025**
Číslo tématu:	**57**		
Název tématu:	**Webová aplikace pro tvorbu 3D map**		
Rozsah práce:	**15 - 25 stránek textu**		

Specifické úkoly, které tato práce řeší:

Cílem této ročníkové práce je vytvořit webovou aplikaci, která umožní procedurálně generovat místností pomocí herního enginu Unity a programovacího jazyka C#. Rozsah, složitost, počet místností a pater těchto vytvořených místnosti bude záviset na uživatelem zadaných parametrech a seedů.

Vytvořené mapy bude možné po dokončení prozkoumat přímo v aplikaci nebo uložit do vlastního souborového formátu (.dnd). Tento formát bude využívat technologie podobné formátu XML.

Termín odevzdání:	**28. března 2025, 23.00**
Vedoucí projektu:	**Bc. Matěj Brožek**
Oponent:	**Ing. Drahomír Škárka**
Schválil:	**Ing. Petra Hrbáčková, ředitelka školy**

ABSTRAKT

Cílem této práce je tvorba a popis tvorby webové aplikace pro procedurální generování map pro TTRPG hry, jako je Dungeons & Dragons, za využití herního enginu Unity a programovacího jazyka C#. Aplikace umožní uživatelům zadávat parametry, jako je počet pater, frekvence a typy dekorativních prvků, přičemž klíčovým faktorem pro generování map bude uživatelem zadaný seed. Procedurálně vytvořené místnosti bude možné prozkoumat přímo v aplikaci nebo exportovat do souboru ve vlastním formátu ".dnd", který bude využívat technologie podobné XML. Pro agilní plánování projektu bude použita metoda Kanban, přičemž správa verzí a dokumentace bude řešena prostřednictvím GitHubu.

KLÍČOVÁ SLOVA

Unity, C#, GameObject, Seed, Procedurálně generováno, D&D

ABSTRACT

The aim of this thesis is to design and describe the creation of a web application for procedural map generation for TTRPG games such as Dungeons & Dragons, using the Unity game engine and the C# programming language. The application will allow users to specify parameters such as the number of floors, frequency and types of decorative elements, with the user-specified seed being the key factor for the map generation. Procedurally created rooms will be able to be explored directly in the application or exported to a custom ".dnd" file using XML-like technologies. Kanban will be used for agile project planning, with version control and documentation handled through GitHub.

KEYWORDS

Unity, C#, GameObject, Seed, Procedurally Generated, D&D

PODĚKOVÁNÍ

Rád bych poděkoval Bc. Matěji Brožkovi za jeho cenné vedení a rady při vypracování mé maturitní práce. Dále děkuji SPŠT a paní ředitelce Ing. Petře Hrbáčkové za tuto unikátní příležitost. V neposlední řadě chci poděkovat platformě itch.io za hosting a věrnou, aktivní komunitu internetových programátorů, kteří jsou vždy ochotni pomoci.

V Třebíči dne 27. března 2025 podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil.

V Třebíči dne 27. března 2025

podpis autora

Obsah

Úvod

Cílem této práce je tvorba webové aplikace pro vytváření map pro TTRPG hry typu Dungeons & Dragons. Aplikace bude využívat herní engine Unity a programovací jazyk C#, hlavně třídu GameObject. Pro agilní plánování projektu bude použita metoda Kanban na platformě Freelo. Správa úložišť, dokumentace a verzí bude zajištěna pomocí systému git, hostovaného na GitHubu.

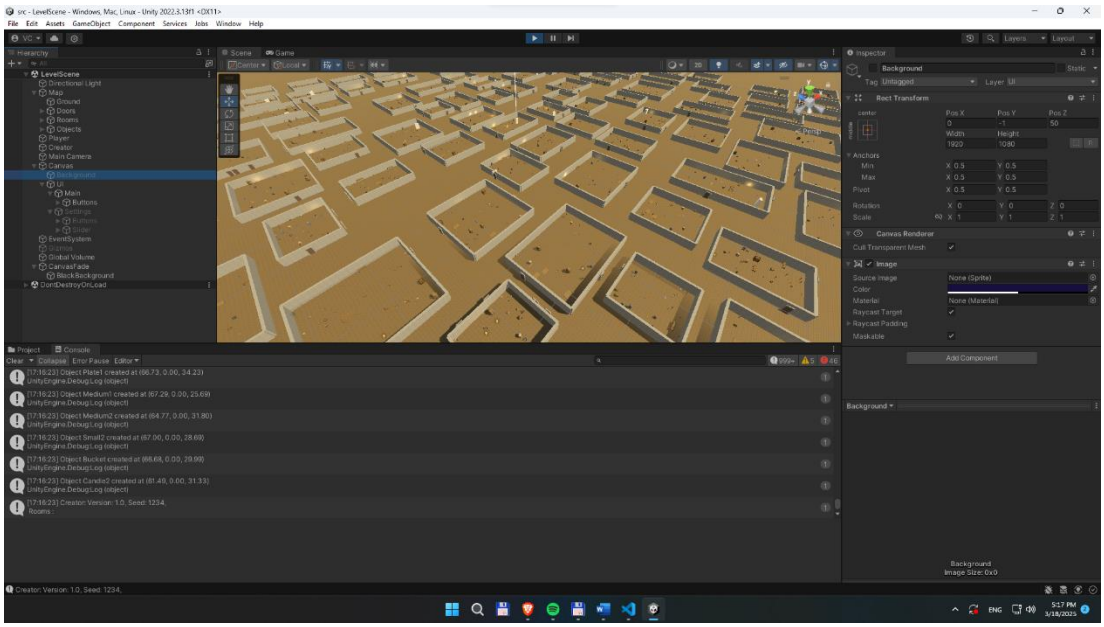
Aplikace bude procedurálně generovat na základě uživatelem zadaných parametrů. Nejvýznamnějším parametrem bude unikátní seed, který bude základem pro generování mapy. Mezi další parametry bude patřit počet pater, četnost a typy dekorativních předmětů, s možností vybrat, které se mají zobrazit a které ne. Vytvořené procedurální místnosti bude možné prozkoumat přímo v aplikaci nebo exportovat do vlastního souborového formátu s příponou „.dnd“. Tento formát bude využívat technologie podobné XML a SVG pro snadné zpracování a čitelnost.

1 Použité technologie

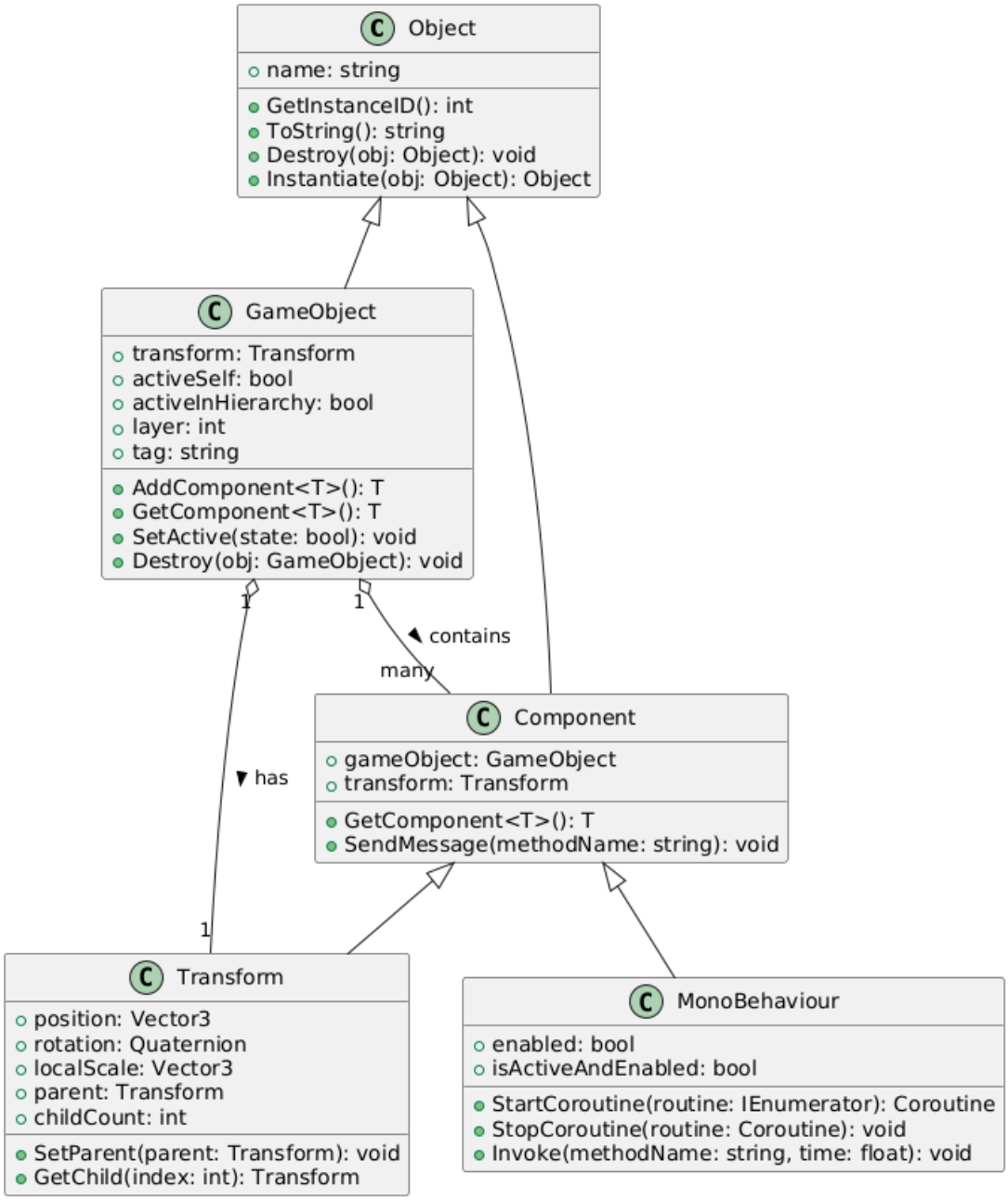
V této části přiblížím, ukážu a vysvětlím technologie tohoto projektu.

1.1 Unity

Unity je zevrubný **herní engine**, který umožňuje vytvářet 2D i 3D hry a interaktivní aplikace. Je známý svou flexibilitou, snadným použitím a širokou podporou platforem. Zakladatel společnosti stojící za Unity ji nazval „*sada nástrojů pro tvorbu her. Je to technologie, která zajišťuje grafiku, zvuk, fyziku, interakce, propojení do sítě*“ [1].



Obrázek 1 Rozložení Unity



Obrázek 2 Základní třídy Unity

Umožňuje programování v jazyce C#, což je efektivní a široce používaný jazyk v herním vývoji. K dispozici jsou také různé nástroje a komponenty pro práci s audiem, síťovými funkcemi, vstupy a dalšími herními prvky.

1.1.1 Object

Object je nejzákladnější třída pro funkčnost a vývoj v Unity. Tato třída by se sama o sobě neměla používat přímo v kódu, protože neobsahuje tolik užitečných metod a

vlastností jako její potomci. Nejvýznamnějšími potomky jsou GameObject, MonoBehaviour a ScriptableObject [2].

Třída Object dále obsahuje metody **Instantiate** a **Destroy**, které slouží k vytváření nebo mazání instancí GameObjectů ve scéně [2].

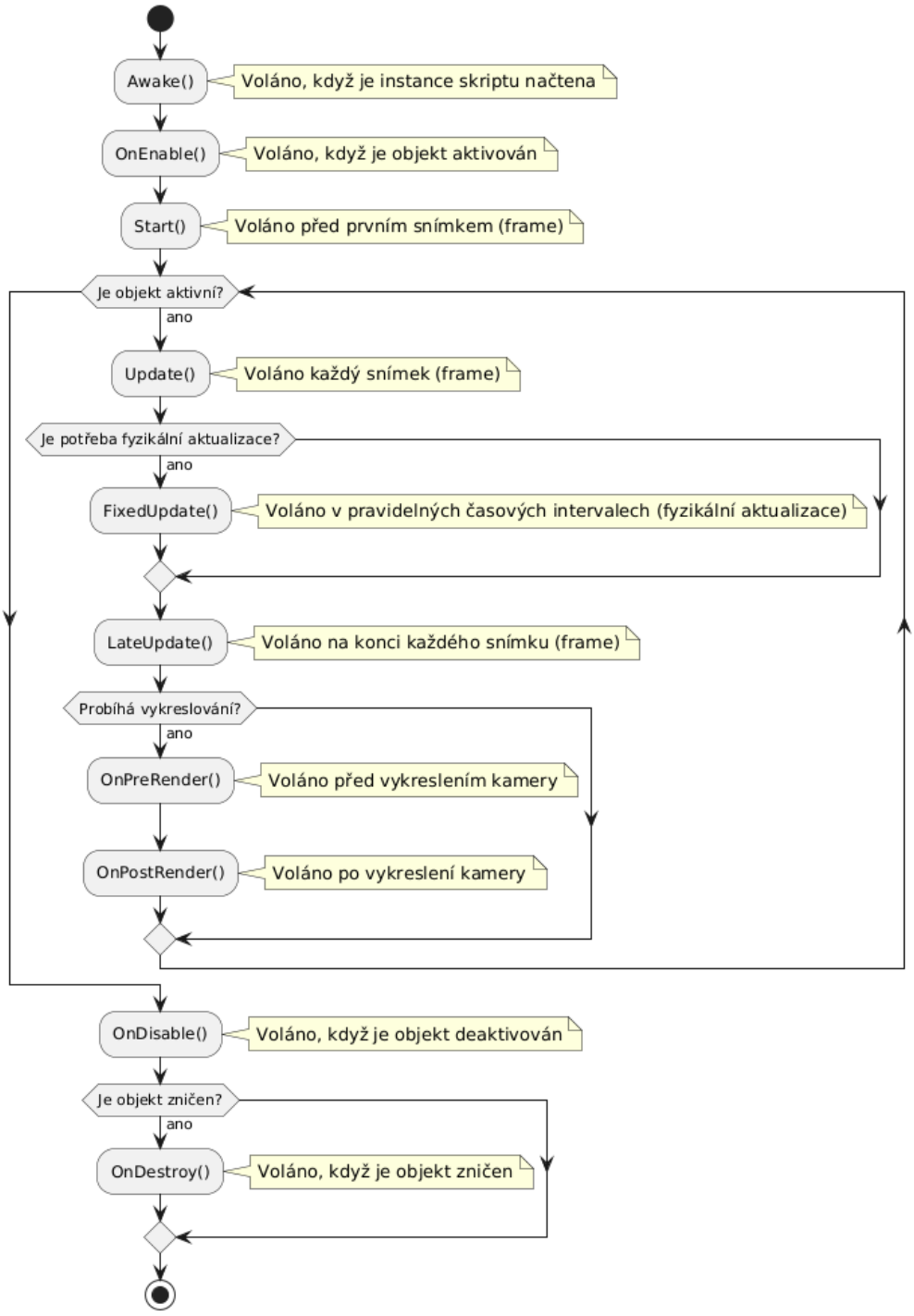
1.1.2 GameObject

GameObject je nejzákladnější třída v Unity pro herní vývojáře. Vše, co se nachází ve scéně Unity musí dědit z této třídy. Poskytuje funkcionality jako: hledání a vytváření odkazů a spojení, zasílání zpráv a proměnných mezi GameObjecty, přidávání, modifikaci a odstraňování **komponentů** a nastavení parametrů a konfigurací [3].

Každý GameObject obsahuje proměnné: název, tag, vrstva, stav a transform. Tyto jednoduché proměnné se využívají k nalezení správné instance ve scéně [3].

1.1.3 MonoBehaviour

Třída MonoBehaviour nám umožňuje připojit skript ke GameObjectu, takže všechny skripty od ní automaticky dědí. Přidává události jako **Awake**, **Update** a **FixedUpdate**. Awake se spustí pouze jednou při vytvoření instance, většinou na začátku scény. Update se spouští každý snímek. FixedUpdate se spouští při každém fyzikálním přepočtu, který je standardně 50krát za sekundu [4].



Obrázek 3 Životní cyklus MonoBehaviour

1.1.4 ScriptableObject

ScriptableObject se používá při ukládání velkých objemů dat do paměti nebo na disk a přenosu dat mezi scénami. Pokud se vytváří několik instancí předem vytvořených GameObjectů, tzv. prefabů, tak ScriptableObject ukládá pouze data,

která se liší od výchozích hodnot. Není možné jej přímo připojit ke GameObjectu, ale musí být uložen jako samostatný soubor, tzv. **Asset** [5].

1.1.5 Transform

Transform je komponenta, která je automaticky přidána ke každému GameObjectu. Poskytuje informace o jeho **pozici**, **rotaci** a **měřítku**. Dále obsahuje metody a proměnné pro jednoduché **hledání potomků** a předků v hierarchii objektů [6].

Existuje alternativa jménem RectTransform, která se hlavně používá pro UI. Lokace v scéně se přes ní určuje pomocí šířky, výšky a pozice obdélníku [7].

1.1.6 TextMesh Pro

TextMesh Pro není základní třídou Unity, ale rozšiřuje možnosti **formátování** a **práce s textem**. Přidává, nebo vylepšuje, také další UI komponenty, jako jsou tlačítka, textová pole, slidery, checkboxy a toggles, které jsou jednodušší na ovládání a optimalizovanější než základní komponenty Unity. Díky tomu je práce s textem a UI v Unity rychlejší a efektivnější [8].

1.2 C#

C# je moderní, **objektově orientovaný programovací jazyk** [3]. Je známý svou jednoduchostí, efektivitou a širokou škálou použití. „*Jazyk C# je odpovědí společnosti Microsoft na jazyk Java a byl vytvořen s cílem být víceúčelovým programovacím jazykem pro všeobecné použití*“ [9].

Výpis 1 C# s prvky Unity

```
private void LateUpdate()
{
    transform.position = player.transform.position + new Vector3(0f, 1f, 0f);

    float fovScale = save.Settings.FOV / 60f;
    float horizontalSensitivity = baseHorizontalSensitivity * fovScale;
    float verticalSensitivity = baseVerticalSensitivity * fovScale;

    Vector2 lookInput = playerInputScript.LookInput;

    if (Mathf.Abs(lookInput.x) > inputThreshold …)
    {
        float mouseX = lookInput.x * horizontalSensitivity * Time.deltaTime;
        float mouseY = lookInput.y * verticalSensitivity * Time.deltaTime;
        …
    }

    camComponent.fieldOfView = save.Settings.FOV;
}
```

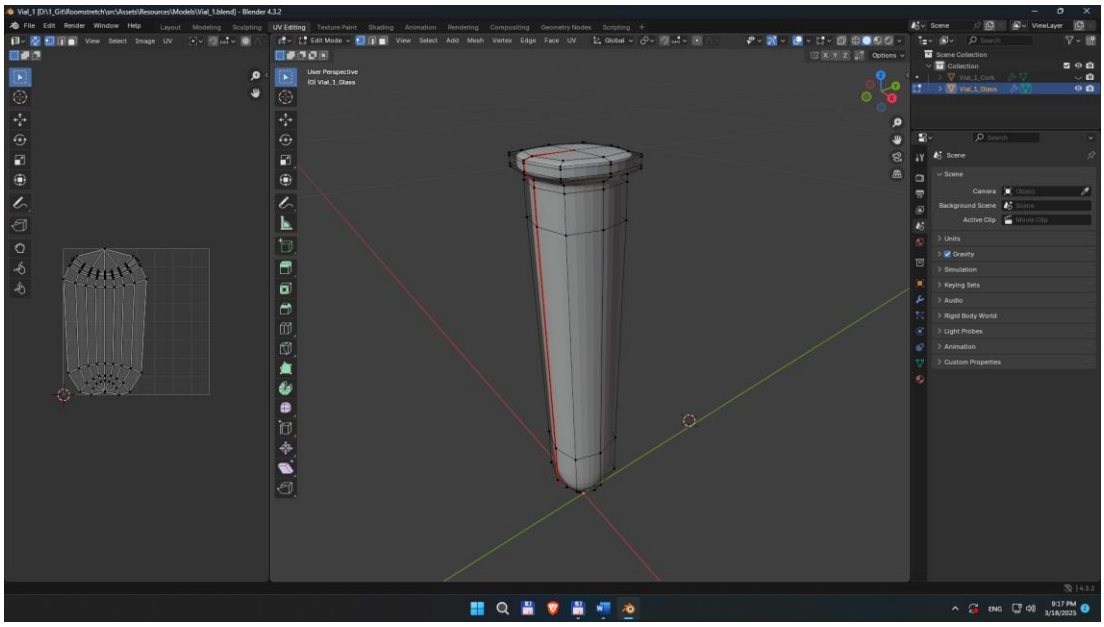
Jazyk je bohatý na funkce, jako jsou **lambda** výrazy, **LINQ** dotazy a **asynchronní** programování, což umožňuje vývojářům psát čistý a efektivní kód. Díky robustní knihovně .NET je C# ideálním nástrojem pro tvorbu multiplatformních aplikací, které běží na různých operačních systémech [10].

1.2.1 Historie

C# vyšel v roce **2002** a od té doby je průběžně aktualizován. Každé 2 roky vychází nová verze. Mezi koncepty, které byly přidány později, patří například lambda výrazy, asynchronní programování a iterátory. Poslední verze rozšířila možnosti práce s parametry a asynchronním programováním [9].

1.3 Blender

Blender je **open-source** software, který slouží k řešení komplexních i jednodušších úloh při tvorbě **3D modelů**. Umožňuje pracovat s topologií, UV unwrappingem, texturami a vytvářením vlastních materiálů. Tento nástroj je ideální pro modelování 3D objektů a texturování, ale také nabízí možnosti pro animaci a rendering. Je oblíbený jak mezi začátečníky, tak profesionály díky své široké škále funkcí [11].



Obrázek 4 UV Unwrapping v Blenderu

1.4 Git

Git je verzovací systém vyvinutý **Linusem Torvaldsem** v roce 2005 [12] při vývoji operačního systému Linux. Git umožňuje sledovat změny v kódu, vracet se k předchozím verzím a spolupracovat s více vývojáři na jednom projektu současně. Tento systém byl vytvořen jako alternativa k předchozím verzovacím nástrojům, které nebyly dostatečně flexibilní pro potřeby open-source vývoje, a od svého vzniku se stal nejrozšířenějším nástrojem pro správu verzí v softwarovém vývoji.

2 Vlastní souborový formát

2.1 XML

XML je formát souboru zapisovaný v plaintextu. Tento formát byl navržen tak, aby byl čitelný jak pro lidi, tak pro stroje. XML je značkovací jazyk, který využívá tzv. tágy k označení struktur a obsahu dat. Tyto tágy nejsou předdefinované, na rozdíl od HTML. XML slouží k přenosu a ukládání dat. [13]

Výpis 2 Vzorový XML Dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<knihovna>
    <kategorie nazev="Programování">
        <kniha id="1" dostupnost="ano">
            <nazev>Unity pro pokročilé</nazev>
            <autor>Karel Malý</autor>
            <rok_vydani>2021</rok_vydani>
        </kniha>

        <upozorneni text="Některé knihy jsou momentálně vypůjčené." />
    </kategorie>
</knihovna>
```

XML soubory obvykle začínají deklarací verze a kódování. Následně obsahují hierarchickou strukturu složenou z otevíracích a uzavíracích tágu a atributu.

2.2 DND

Stejně jako jeho „předchůdce“ XML je i formát .dnd značkovací jazyk využívající tágy. Název tohoto souborového formátu je inspirován hrou Dungeons & Dragons, pro kterou je tento projekt určen.

Formát .dnd byl navržen specificky pro potřeby tohoto projektu, aby umožňoval snadnou úpravu herního obsahu pomocí jednoduché syntaxe.

2.2.1 Struktura

Na rozdíl od XML souboru používá DND strukturu hlavy (head) a těla (body). Výhodou této struktury je oddělení metadat od dat. To nám zajišťuje lepší přehlednost a organizaci souboru.

Výpis 3 Struktura DND souboru

```
<?xml version="1.0" encoding="utf-8"?>
<RoomStretch>
  <Head>
    <Save>
      …
    </Save>
    <Settings>
      …
    </Settings>
  </Head>
  <Body>
    <Rooms>
      …
    </Rooms>
    <Doors>
      …
    </Doors>
    <Objects>
      …
    </Objects>
  </Body>
</RoomStretch>
```

Tato struktura také umožňuje snadnou rozšiřitelnost. Zjednodušuje strojové i lidské ctění.

2.3 Head

V části head DND souboru se nachází všechny parametry určené pro celý soubor, jako je seed generátoru, verze programu a další uživatelem zadané proměnné.

2.3.1 Body

V části body jsou uloženy informace o herním prostředí, především o místnostech, jejich velikosti, pozici a stavu použití vygenerovaných prvků.

3 Table Top Role-Playing Games

Table Top Role-Playing Games, dále jen TTRPG, neboli stolní hry na hrdiny, je žánr fantasy a sci-fi her. Hráči se v nich vžívají do rolí hrdinů, kteří tvoří družinu dobrodruhu ve fantaskním světě tvořeným jedním z nich. Speciální hráč, většinou přezdívaný Pán Jeskyně nebo Dungeon Master, dále jen PJ, tyto příběhy, postavy a někdy i cele světy připravuje sám. To může být náročné, tudíž existuje mnoho online i offline nástrojů, které s některými repetetivními částmi pomáhají. Tento projekt se řadí mezi tyto nástroje [14].

3.1 Historie

"*Kdysi velmi, velmi dávno v říši zvané Středozápad Spojených států, konkrétně ve státech Minnesota a Wisconsin, se scházela skupina přátel, aby navždy změnila herní historii*" [14] píše Mike Mearls, vedoucí návrhu D&D. Hlavními tvůrčími osobnostmi byli přátelé Mike Mearls a Jeremy Crawford, kteří byli frustrovaní ne-interaktivními příběhy z knih a filmu. Tudíž se rozhodli vytvořit si nový žánr her, kde se nikdy nestane to stejně dvakrát.

"*Dungeons & Dragons odstartovalo celosvětový fenomén. Je to první hra na hrdiny a zůstává nejlepší svého druhu*" [14] píše Mearls.

3.2 Verze

DnD je pouze jedna z variant tohoto populárního žánru. DnD samotné má 5 edicí, kdy každá má svoji oddanou komunitu. Pro více komplexní fantasy zážitek existují Pathfinder a jeho edice [15]. Neexistují pouze fantasy TTRPG hry, velmi populární jsou sci-fi verze jako Traveler, Starfinder nebo Cyberpunk. Hororové témata jsou taktéž hojná jako třeba Mörk Borg nebo Call of Cthulhu [16].

3.3 Mechaniky

Většina stolních her na hrdiny má svá komplexní pravidla a mechaniky, ale vše je založeno na 20 stěnné kostce [14]. Tato kostka, nebo jiné velikosti, určuje úspěch nebo neúspěch při různých akcích, jako je boj, hádanky nebo vyjednávání. Hráči mají volnou ruku nad prozkoumáváním světa, kde hledají magické a jinak užitečné předměty.

Každý systém má vlastní pravidla, která určují míru realismu a hloubku hry. Například Dungeons & Dragons klade důraz na hrdinský boj, Mörk Borg se zaměřuje na smrt a pokus o přežití v apokalyptickém světě [16].

4 Scény

Aplikace je rozdělena do 3 základních scén, které fungují nezávisle na sobě a každá plní specifickou funkci v rámci aplikace. Každá scéna má svůj vlastní účel, logiku a vizuální reprezentaci, což umožňuje modularitu a snadnou údržbu kódu.

4.1 MainMenuScene

MainMenuScene slouží jako úvodní obrazovka aplikace, kde uživatel interaguje s hlavním menu. Tato scéna je klíčová pro prvotní uživatelskou zkušenost, protože uživateli poskytuje možnosti, jako je začátek nové hry, a přístup k nastavení.

Funkce ukončení aplikace se zde nenachází, protože je zbytečné aplikaci hostovanou skrz WebGL vypínat. Stačí zavřít okno prohlížeče.

Při načtení scény se **inicializují** všechny potřebné proměnné a nastavení, aby byla zajištěna správná funkčnost menu. Uživatel může pomocí tlačítek procházet jednotlivé možnosti menu. Každé tlačítko spouští specifickou akci přechod: do další scény nebo otevření nastavení a navrácení zpět. V **nastavení** může uživatel upravit základní parametry, například hlasitost zvuku a FOV (Field Of View).

Obrázek 5 MainMenuScene

Scéna využívá UI komponenty Unity, tlačítka, textová pole a slidery, pro vytvoření interaktivního menu. Pro plynulé přechody mezi scénami se používá **SceneManager** z Unity.

4.2 GeneratingScene

GeneratingScene je scéna, ve které probíhá procedurální generování obsahu na základě uživatelem zadaných parametrů. Tato scéna je klíčová pro vytvoření dynamického a přizpůsobitelného herního světa.

Uživatel zadává různé **parametry**, velikost mapy a místností, počet místností a objektů. Tyto vstupy jsou validovány a uloženy do odpovídajících datových struktur. Na základě zadaných parametrů se generuje struktura mapy, umístění objektů a další herní elementy. Tento proces využívá algoritmy pro náhodné generování, jako je vzorkování s odmítáním, aby byla zajištěna variabilita a zajímavost herního světa.

Obrázek 6 GeneratingScene

Vygenerovaná data jsou **uložena** do souboru, která se později používá v LevelScene.

4.3 LevelScene

LevelScene je dynamicky generovaná scéna, která se vytváří při spuštění aplikace na základě uložených dat (**DNDFileData**). Tato scéna je klíčová pro samotnou hru, protože obsahuje herní úroveň, se kterou uživatel interaguje.

Scéna neobsahuje žádný statický obsah. Mapa, objekty a herní elementy jsou načteny z uložených dat, což umožňuje flexibilitu a přizpůsobení se potřebám uživatele. Uživatel zde může prozkoumávat herní svět.

Scéna využívá dynamické načítání assetů, jako jsou prefaby, textury a modely, z uložených dat. Pro interakci s objekty se používají fyzikální komponenty Unity, jako jsou **Colliders** a **Rigidbody**.

5 Důležité skripty

Aplikace má několik skriptu a tato část ukáže ty **nejdůležitější**, které stojí za hlavní funkčností a generováním.

5.1 UIScript

UIScript je klíčový skript, který zastřešuje veškeré uživatelské rozhraní (UI) v projektu. Tento skript je zodpovědný za dynamické ovládání UI elementů v různých scénách, propojení UI s herní logikou a zajištění interaktivity mezi uživatelem a aplikací.

V **MainMenuScene** UIScript zajišťuje základní navigaci mezi možnostmi menu, jako je začátek nové hry a přístup k nastavení. Skript také inicializuje potřebné proměnné a nastavení, aby bylo menu plně funkční a připravené pro interakci s uživatelem.

V **GeneratingScene** dynamicky zapíná a vypíná UI elementy na základě uživatelských vstupů. Například, pokud uživatel nechce zadávat vlastní hodnoty pro počet místností nebo objektů, skript deaktivuje příslušná vstupní pole a použije výchozí hodnoty. Skript také zpracovává uživatelské vstupy, jako jsou parametry pro generování mapy (např. velikost mapy, počet místností, počet objektů) a ukládá je do datové struktury DNDFileData.

V **LevelScene** ovládá herní menu a nastavení. Skript zajišťuje, aby se menu zobrazilo nebo skrylo na základě uživatelských akcí (Esc tlačítko). Propojuje nastavení v menu s herními hodnotami, jako je FOV (Field of View) nebo senzitivita pohybu kamery. Dále UIScript ovládá přepínání mezi herními a menu akcemi pomocí Input Action Maps. Když je menu otevřené, herní ovládání je deaktivováno, a naopak.

5.1.1 Metody

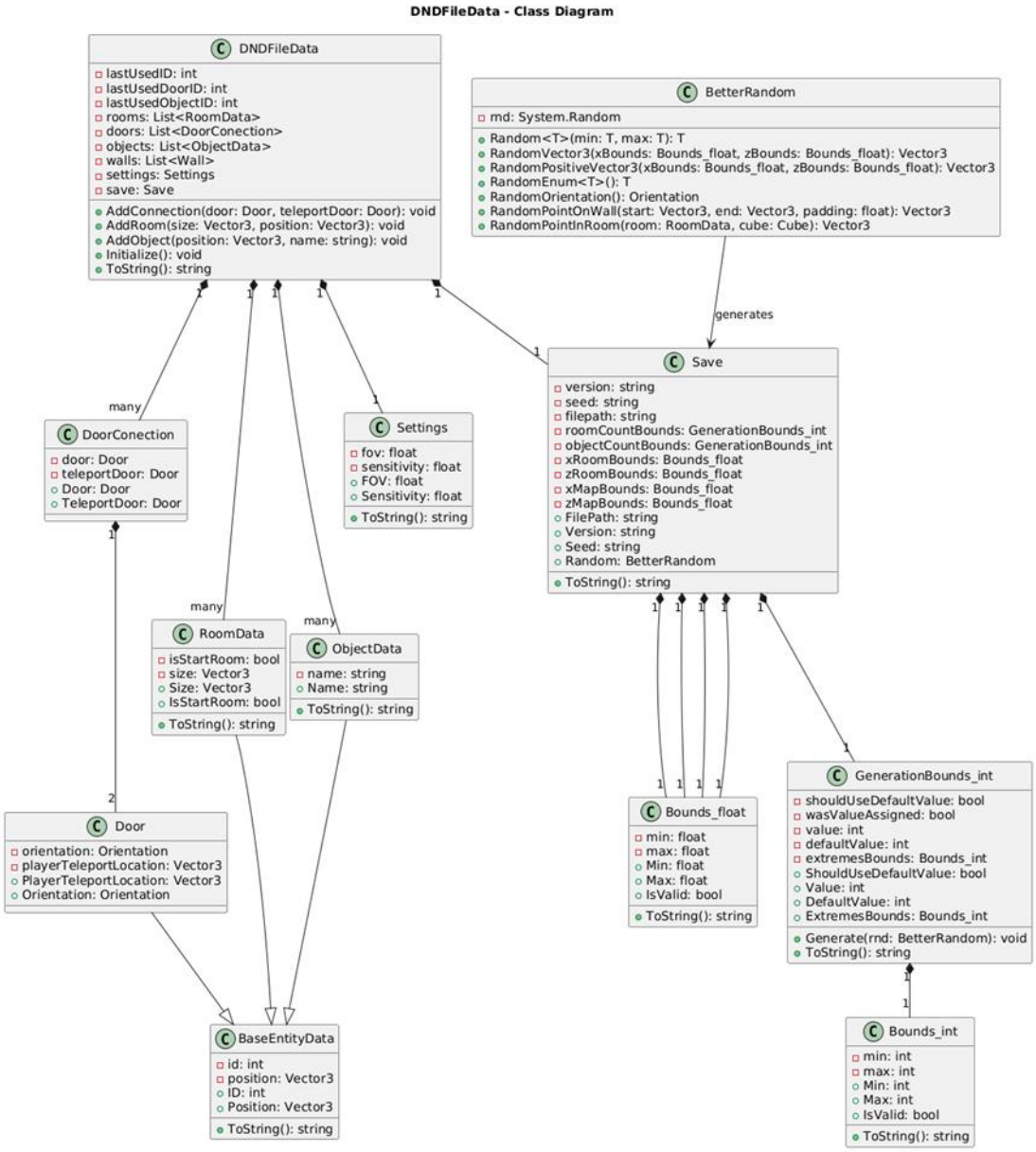
SubmitButton_Click zpracovává uživatelské vstupy z GeneratingScene, ukládá je do datové struktury DNDFileData a spouští generování mapy. Poté přechází na další scénu.

NextScene() a **PreviousScene()** slouží k přechodu mezi scénami. NextScene načte následující scénu, zatímco PreviousScene načte předchozí scénu. Vždy nastaví správnou scénu na aktivní, aby nedocházelo k problémům s přechody.

LockCursor(bool shouldBeLocked) ovládá viditelnost a stav kurzoru myši. Když je kurzor uzamčen, uživatel může ovládat herní kameru, a když je odemčen, může interagovat s UI.

5.2 DNDFileData

DNDFileData je klíčový skript, který obsahuje několik pomocných tříd pro ukládání dat a práci s nimi. Tyto třídy slouží k reprezentaci herních objektů, nastavení a parametrů pro procedurální generování. Skript je navržen tak, aby byl modulární a snadno rozšiřitelný. Všechny třídy jsou označeny atributem **[System.Serializable]**, což umožňuje jejich ukládání a načítání pomocí Unity serializace. Díky abstraktní třídě BaseEntityData je snadné přidávat nové typy objektů bez nutnosti změny stávajícího kódu.



Obrázek 7 Vizualizace DNDFileData

DNDFileData obsahuje klíčové metody na přidávání objektu. **AddRoom()** přidá novou místnost do seznamu místností. **AddObject()** přidá nový objekt do seznamu objektů. **AddConnection()** přidá propojení mezi dvěma dveřmi. **Initialize()** inicializuje všechny seznamy a proměnné na výchozí hodnoty.

5.2.1 BaseEntityData

Tato abstraktní třída je základem pro všechny ukládané objekty ve scéně. Obsahuje společné vlastnosti, jako je ID (identifikátor objektu) a Position (pozice objektu ve scéně).

Třídy RoomData, Door a ObjectData dědí od BaseEntityData a rozšiřují ji o specifické vlastnosti pro každý typ objektu.

5.2.2 RoomData

Reprezentuje místnost ve scéně. Kromě základních vlastností z BaseEntityData obsahuje také Size (velikost místnosti) a IsStartRoom (indikátor, zda je místnost výchozí místností). Tato třída se používá pro ukládání informací o místnostech generovaných v GeneratingScene.

5.2.3 DoorConection a Door

DoorConection reprezentuje propojení mezi dvěma dveřmi (dveře a jejich teleportační protějšky).

Door obsahuje informace o pozici dveří, jejich orientaci (Orientation) a pozici, na kterou se hráč teleportuje (PlayerTeleportLocation).

5.2.4 ObjectData

Reprezentuje objekty ve scéně, jako jsou světla, nábytek nebo dekorace. Kromě základních vlastností z BaseEntityData obsahuje také Name (název objektu) a výčtové typy (TypesOfObjects, LightTypes, FurnitureTypes atd.), které specifikují typ objektu.

Tato třída se používá pro ukládání informací o objektech umístěných v místnostech.

5.2.5 Settings a Save

Settings uchovává nastavení aplikace, jako je FOV a citlivost ovládání. Nastavení lze upravovat v menu a jsou uložena pro pozdější použití.

Save obsahuje data zadaná uživatelem při generování místností, jako je Seed (semínko pro náhodné generování), RoomsCountBounds (rozsah počtu místností), ObjectCountBounds (rozsah počtu objektů) a Bounds pro rozměry místností a mapy.

5.2.6 GenerationBounds a Bounds

GenerationBounds je generická třída, která slouží k ukládání rozsahů pro generování náhodných čísel. Obsahuje vlastnosti jako ShouldUseDefaultValue (použít výchozí hodnotu), Value (aktuální hodnota) a ExtremesBounds (rozsah hodnot).

Bounds je generická struktura, která definuje minimální a maximální hodnoty pro daný typ (např. int, float).

5.2.7 BetterRandom

Tato třída rozšiřuje funkcionalitu standardní třídy **Random** z namespace System. Obsahuje metody pro generování náhodných čísel (Random) a náhodných vektorů (RandomVector3). RandomVector3 je přetížená pro 2D a 3D. 2D verze se používá pro generování místností, zatímco 3D verze se používá pro generování objektů.

5.3 Creator

Creator je prefab, který ovládá generovaní a převádění dat do souboru a datových struktur. Obsahuje dvě třídy: DNDFileScriptCreator a DNDSceneScriptCreator.

5.3.1 DNDFileScriptCreator

DNDFileScriptCreator obsahuje dvě hlavní části, a to metodu PrepareSave a CreateFile s pomocnými metodami. Slouží k vytváření souboru.

PrepareSave stojí za samotnou generací dat. Využívá podpůrné metody na vytvoření místností. Poté je propojí pomocí pomocných tříd **DoorConection** a **Door**. Místnosti naplní objekty, při tom používá třídu **Cube**, aby se objekty nepřekrývaly.

PlaceRoom, **PlaceDoor** a **PlaceObject** využívají vzorkování s odmítáním na generaci zvolených datových struktur. Vždy zkouší 100x vygenerovat datovou strukturu, pokud se podmínka nesplní tak přeskakuje objekt a posouvá se dál.

CreateFile používá třídu **XmlWriter** na zapisování do souboru. Tato třída je intuitivní a lehce rozšiřitelná.

Výpis 4 Práce s XmlWriter

```
private void WriteVector3(XmlWriter writer, Vector3 position, string name)
{
    writer.WriteStartElement(name);

    writer.WriteElementString("X", position.x.ToString());
    writer.WriteElementString("Y", position.y.ToString());
    writer.WriteElementString("Z", position.z.ToString());

    writer.WriteEndElement();
}
```

Metody **WriteGenerationBounds**, **WriteBounds** a **WriteVector3** odstraňují duplicitní kód při psaní opakovaných datových struktur.

5.3.2 DNDSceneScriptCreator

DNDSceneScriptCreator obsahuje metody na načítání dat, převádění souborů do datové struktury a vytvoření scény samotné.

Data načítá pomoci Unity třídy **Resources**. Aby se tato třída mohla použít tak chtěná data musí být ve složce Assets/Resources. Takto načtená data se ukládají do slovníku, aby se jednoduše a intuitivně používala v jiných metodách.

MakeMap a převede datovou strukturu **DNDFileData** do scény. Využívá Unity metodu **Instantiate** s přetížením pro rodiče, aby se zachovala logická hierarchie.

pomocné metody **InstantiateRoom**, **InstantiateDoor** a **InstantiateObject** vytváří samotný obsah scény. Pomoci výčtového typu Orientation správně otáčí zdi a dveře aby vždy směrovaly k centru místnosti.

Výpis 5 Pomocná parsovaci metoda na GenerationBounds

```
    private GenerationBounds<T> ParseGenerationBounds<T>(XElement
element) where T : IComparable<T>
    {
        bool ShouldUseDefaultValue =
            element.Element("ShouldUseDefaultValue").Value.Trim() == "True";
        T value =
            (T)Convert.ChangeType(element.Element("Value").Value.Trim(),
                                  typeof(T));
        T defaultValue =
            (T)Convert.ChangeType(element.Element("DefaultValue").Value.Trim
(),
                                  typeof(T));
        Bounds<T> extremesBounds =
            ParseBounds<T>(element.Element("ExtremesBounds"));

        GenerationBounds<T> generationBounds =
            new GenerationBounds<T>(defaultValue,
                                    extremesBounds);
        generationBounds.ShouldUseDefaultValue = ShouldUseDefaultValue;
        generationBounds.Value = value;

        return generationBounds;
    }
```

Metoda **ParseDNDFile** převede **.dnd** soubor zpět do datové struktury, aby se s ní dalo jednoduše manipulovat. Pomocné parsovací metody odstraňují duplicitní kód a zjednodušují psaní programu a možnou rozšiřitelnost.

5.4 MaterialTilingUpdater

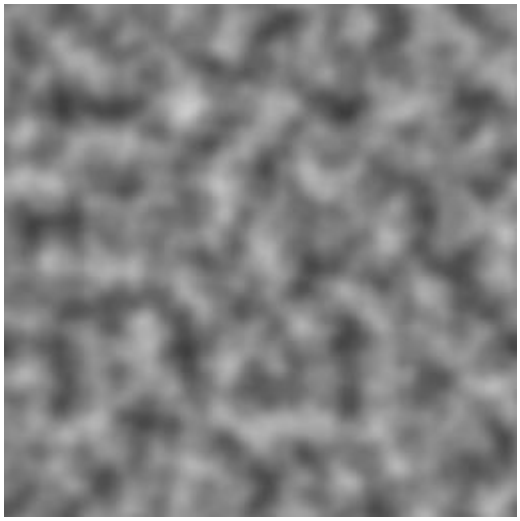
MaterialTilingUpdater je krátký, ale důležitý skript, který řeší problém s vizuálním zobrazením textur na zdech místností. Při procedurálním generování místností se zdi natahují na požadovanou délku, což může způsobit deformaci nebo nekonzistentní zobrazení 2D textur (materiálů). Tento skript zajišťuje, že textury na zdech vypadají jednotně a esteticky, bez ohledu na jejich velikost.

6 Procedurální generace

Procedurální generování je klíčovým mechanismem aplikace, který umožňuje dynamické a náhodné vytváření herních úrovní a map. Tento proces využívá algoritmy a matematické funkce k vytvoření herních prvků, jako jsou terény, objekty a struktury, na základě parametrů zadaných uživatelem.

6.1 Příklady

Jedním z nejznámějších příkladů použití procedurálního generování je hra Minecraft, která využívá techniku Perlinova šumu k vytváření realistických a přírodních terénů. Perlinův šum je matematická funkce, která generuje plynulé a přirozeně vypadající vzory, což se ideálně hodí pro generování hor, řek, lesů a dalších přírodních prvků. Tento typ šumu je základem pro vytváření "náhodného" světa, který vypadá konzistentně a realisticky, i když je generován automaticky.



Obrázek 8 Perlinův šum (http://kitfox.com/projects/perlinNoiseMaker/)

Tento přístup umožňuje generování neomezeného množství unikátních map, což zvyšuje replay value a herní hodnotu. Šetří prostor v paměti, protože není potřeba uchovávat celou mapu – místo toho se vždy generuje nová na základě aktuálních podmínek.

Procedurální generování může být aplikováno na různé aspekty hry, od terénu, přes umístění objektů až po chování NPC postav, což dává hráčům novou zkušenost s každým hraním

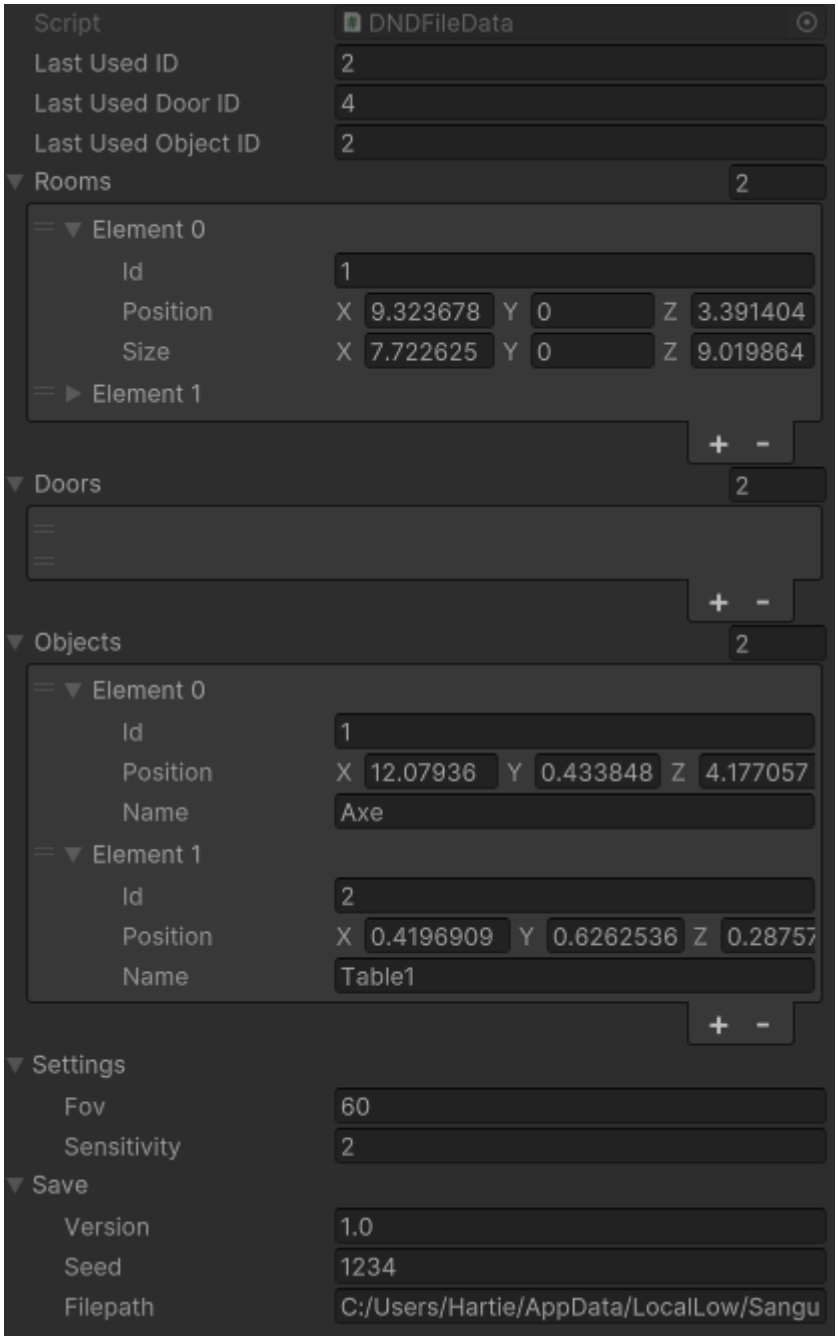
6.2 Proces

Procedurální generování v této aplikaci začíná příjmem dat od uživatele v **GeneratingScene**, kde se definují požadované vlastnosti mapy. Na základě těchto parametrů se provádí výpočet a generování prostředí, což zahrnuje jak náhodné, tak i určité predikabilní prvky pro zajištění vyváženosti.

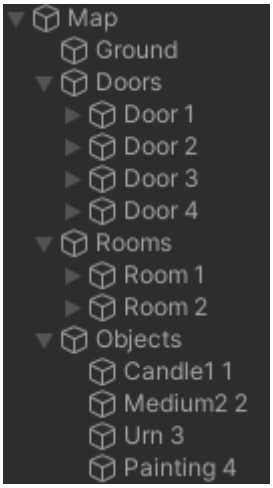
Proces generování začíná vytvořením **místností** pomocí **vzorkování s odmítáním** (rejection sampling). Tato metoda zajišťuje, že místnosti jsou umístěny náhodně, ale zároveň splňují určité podmínky, jako je například to, že se nepřekrývají. Každá místnost má definovanou velikost a pozici, které jsou generovány na základě uživatelem zadaných parametrů, jako je rozsah velikostí místností.

Po vygenerování místností jsou propojeny pomocí **DoorConnection**. Každá dvojice dveří (vstup a výstup) je propojena tak, aby vytvořila logickou cestu mezi místnostmi. Tento krok zajišťuje, že mapa je propojená a hráč se může volně pohybovat mezi místnostmi.

Následně jsou místnosti zaplněny **objekty**, které jsou rozděleny do čtyř **výčtových typů**: Light (světla), Furniture (nábytek), Wall (stěny) a Decoration (dekorace). Každý typ objektu má stejnou pravděpodobnost výskytu, ale jednotlivé typy obsahují různé množství objektů. Například pravděpodobnost výskytu dekorací je vyšší než pravděpodobnost výskytu konkrétního kusu nábytku, jako je stůl. Objekty jsou umisťovány tak, aby se nepřekrývaly, což je zajištěno pomocí třídy **Cube**, která kontroluje kolize mezi objekty.



Obrázek 9 DNDFileData zobrazené v Unity Inspektoru



Obrázek 10 Výsledná Hierarchie

Po dokončení generování je vytvořena **logická hierarchie** objektů ve scéně, která zahrnuje místnosti, dveře a objekty. Tato hierarchie je uložena v datové struktuře DNDFileData a může být později použita pro načtení scény.

7 Post Processing

Post Processing je sada efektů, které se aplikují na již vyrenderovanou scénu, aby upravily její vizuální vlastnosti. Tyto efekty se používají k dosažení vyšší vizuální kvality, filmového nebo herního vzhledu, nebo k celkovému stylizování scény. Post Processing může výrazně ovlivnit atmosféru a náladu hry, a proto je důležitou součástí moderního herního vývoje.

7.1 Běžné efekty v Post Processingu

Mezi nejčastěji používané post-processing efekty patří **Bloom**, který zvyšuje jas světlých částí scény a vytváří efekt záře, **Depth of Field**, který rozmazává objekty v popředí nebo pozadí, aby simuloval hloubku ostrosti, a **Color Grading**, který upravuje barevnou paletu a kontrast scény. Další běžné efekty zahrnují **Motion Blur** pro rozmazání pohybu, **White Balance** pro úpravu teploty světla a **Antialiasing**, který vyhlazuje ostré hrany.

7.2 Používané efekty

V aplikaci jsou použity následující post-processing efekty: **Tonemapping**, který upravuje rozsah jasu ve scéně, White Balance pro úpravu teploty světla, **Depth of Field** pro vytvoření hloubky ostrosti, **Lift Gamma Gain** pro přesné nastavení barev a světlosti, **Motion Blur** pro rozmazání pohybu a **Bloom** pro zajištění záře kolem světelných zdrojů. Tyto efekty společně přispívají k celkovému vizuálnímu zážitku a atmosféře hry.

7.3 Výhody použití Post Processingu

Post Processing přináší několik výhod, jako je vylepšení vizuální kvality, vytvoření specifické atmosféry a zvýšení realističnosti scény. Efekty jako Bloom, Depth of Field a Color Grading umožňují dosáhnout požadovaného vizuálního stylu a zlepšit celkový dojem z hry.

8 Klíčové algoritmy

8.1 Vzorkování s odmítáním

Vzorkování s odmítáním, anglicky **Rejection Sampling**, je algoritmus na procedurální generovaní platných dat. Funguje tak, že generuje náhodné hodnoty (např. polohy místností) a ty, které nesplňují podmínky (např. překrývají se s jinými místnostmi), odmítne a zkouší to znovu. Tento proces opakuje, dokud nenajde vhodnou hodnotu

Aplikace tento algoritmus používá ke generovaní místnosti, dveří a objektů do DNDFileData.

8.2 Kontrola kolizí

Kontrola kolizí je algoritmus používaný k detekci překryvů mezi objekty. V aplikaci je tento algoritmus klíčový pro umisťování objektů a zajištění, že se nepřekrývají.

Při generování objektů v místnostech se kontroluje, zda se nově umístěný objekt nepřekrývá s jinými objekty nebo stěnami. Pokud ano, objekt je odmítnut a generování se opakuje. Stejný princip platí pro umisťování dveří, které nesmí blokovat průchody nebo překrývat jiné prvky scény.

Výpis 6 Kontrola kolizí třídy Cube

```
public bool Intersects(Cube other)
{
    return (position.x - size.x) < (other.position.x + other.size.x) &&
           (position.x + size.x) > (other.position.x - other.size.x) &&
           (position.y - size.y) < (other.position.y + other.size.y) &&
           (position.y + size.y) > (other.position.y - other.size.y) &&
           (position.z - size.z) < (other.position.z + other.size.z) &&
           (position.z + size.z) > (other.position.z - other.size.z);
}
```

Výhodou kontroly kolizí je její schopnost zajistit, že objekty jsou umístěny správně a nezpůsobují problémy. Tento algoritmus se snadno integruje s dalšími technikami, jako je vzorkování s odmítáním.

8.3 Serializace a deserializace dat

Serializace a deserializace jsou algoritmy používané k ukládání a načítání dat. V aplikaci jsou tyto algoritmy klíčové pro ukládání generovaných map a jejich pozdější načítání.

Při generování mapy jsou data serializována do souborů, aby mohla být později načtena. Serializace zahrnuje převod datových struktur, jako jsou místnosti, dveře a objekty, do formátu, který lze uložit do souboru. Při načítání mapy jsou data deserializována a převedena zpět do datových struktur, se kterými může aplikace pracovat.

Výhodou serializace a deserializace je jejich schopnost zachovat komplexní datové struktury a zajistit konzistenci a integritu dat. Tyto algoritmy umožňují ukládání a načítání map bez ztráty informací.

8.4 Fyzikální simulace

Fyzikální simulace je klíčová pro realistické chování objektů ve scéně. V aplikaci je použita k simulaci pohybu hráče.

Collidery a Rigidbody zajišťují, že objekty správně interagují s prostředím a navzájem se neprostupují. Tyto algoritmy se používají k pohybu objektů a jejich interakcí.

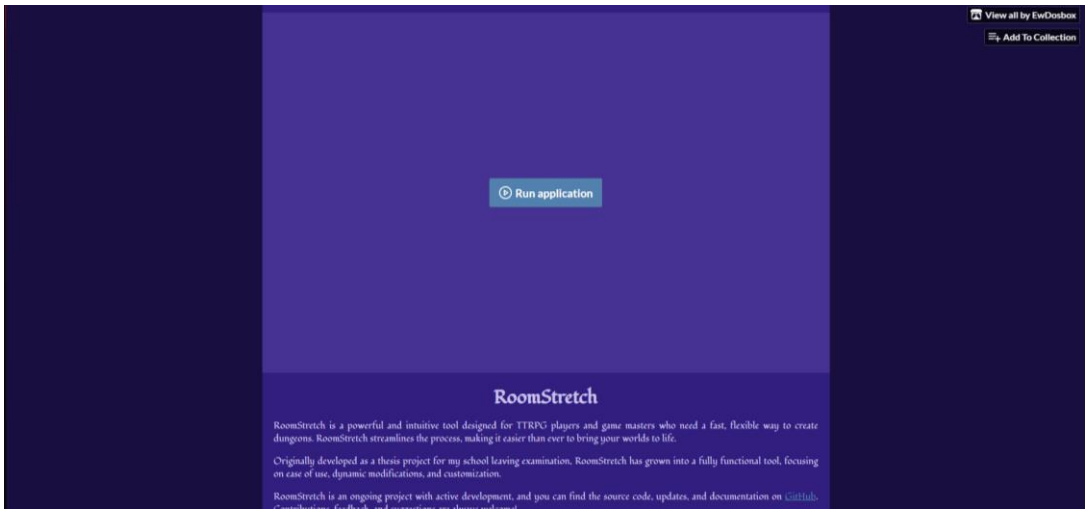
9 Port na web

9.1 WebGL

Projekt využívá technologii WebGL, což je technologie pro vykreslování 3D grafiky přímo v prohlížeči bez nutnosti instalace jakýchkoli doplňků. WebGL je založen na OpenGL ES 2.0 a umožňuje vykreslovat 3D objekty a animace ve webových prohlížečích, což znamená, že aplikace bude přístupná přímo přes internet bez potřeby externího softwaru. Tento přístup je ideální pro interaktivní aplikace a hry, které mají běžet na široké škále zařízení.

9.2 Hostování

Pro tento projekt bude použita platforma itch.io, která umožňuje vývojářům snadno hostovat a distribuovat své hry přímo na internetu. Itch.io podporuje nahrávání her v různých formátech, včetně HTML5 her, které využívají WebGL pro grafiku a interaktivitu. Projekt bude tedy exportován jako HTML5 aplikace, což zaručí kompatibilitu napříč prohlížeči bez potřeby stahování klienta.



Obrázek 11 Itch.Io

Po exportu bude aplikace nahrána na adresu ewdosbox.itch.io\roomstretch, kde ji budou moci uživatelé jednoduše přistupovat a pouzivat přímo v prohlížeči. Itch.io také nabízí nástroje pro nastavení verzí hry, což umožňuje jednoduše aktualizovat projekt bez nutnosti měnit URL.

Závěr

Tato práce se zaměřila na návrh a implementaci webové aplikace pro procedurální generování herních map a scén dle uživatelských parametrů pro stolní hry na hrdiny. Byly využity moderní technologie, jako Unity, C# a algoritmus vzorkování s odmítáním, čímž se podařilo vytvořit flexibilní systém generování herního obsahu. Avšak tento algotimus ma své limitace pro vytváření více komplexních tvarů a objektů.

Hlavním přínosem je unikátnost generovaných map s logickou strukturou místností a objektů, což zajišťují zmíněné algoritmy. Implementovaný systém serializace a deserializace umožňuje ukládání a načítání map, což zvyšuje uživatelskou přívětivost a moznost vytvareni vlastnich map. Důraz byl kladen i na vizuální stránku díky efektum a post-processingu (Bloom, Depth of Field, Color Grading), které dodávají scénám atmosféru a styl.

Celkově projekt demonstruje efektivní využití procedurálních metod pro dynamické generování obsahu ve hrách.

Seznam použitých zdrojů

[1] HAAS, John. *A History of the Unity Game Engine*. Online, . Worcester: WORCESTER POLYTECHNIC INSTITUTE, 2014. Dostupné z: http://www.daelab.cn/wp-content/uploads/2023/09/A_History_of_the_Unity_Game_Engine.pdf. [cit. 2025-01-14].

[2] UNITY TECHNOLOGIES. *Fundamental Unity types*. Online. UNITY TECHNOLOGIES. Unity 6 User Manual. 2025. Dostupné z: https://docs.unity3d.com/Manual/fundamental-unity-types.html. [cit. 2025-01-14].

[3] UNITY TECHNOLOGIES. *GameObject*. Online. UNITY TECHNOLOGIES. Unity 6 User Manual. 2025. Dostupné z: https://docs.unity3d.com/Manual/class-GameObject.html. [cit. 2025-01-14].

[4] UNITY TECHNOLOGIES. *MonoBehaviour*. Online. UNITY TECHNOLOGIES. Unity 6 User Manual. 2025. Dostupné z: https://docs.unity3d.com/Manual/class-MonoBehaviour.html. [cit. 2025-01-14].

[5] UNITY TECHNOLOGIES. *Scriptable Object*. Online. UNITY TECHNOLOGIES. Unity 6 User Manual. 2025. Dostupné z: https://docs.unity3d.com/Manual/class-ScriptableObject.html. [cit. 2025-01-23].

[6] UNITY TECHNOLOGIES. *Transform*. Online. UNITY 6 USER MANUAL. Unity Technologies. 2025. Dostupné z: https://docs.unity3d.com/Manual/scripting-transform.html. [cit. 2025-01-23].

[7] UNITY TECHNOLOGIES. *RectTransform*. Online. UNITY 6 USER MANUAL. Unity Technologies. 2025. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/RectTransform.html. [cit. 2025-01-16].

[8] UNITY TECHNOLOGIES. *TextMesh Pro Documentation*. Online. UNITY TECHNOLOGIES. Unity Documentation. 2025. Dostupné

z: https://docs.unity3d.com/Packages/com.unity.ugui@2.0/manual/TextMeshPro/. [cit. 2025-01-13].

[9] FRIEDMAN, Janice. *When Was C# Created? A Brief History*. Online. C# Station. 2020. Dostupné z: https://csharp-station.com/when-was-c-sharp-created-a-brief-history/. [cit. 2025-01-22].

[10] MICROSOFT. *Prohlídka jazyka C#*. Online. MICROSOFT. Microsoft Learn. 2024. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/overview. [cit. 2025-01-13].

[11] BLENDER. *About Blender*. Online. BLENDER. Blender 4.3 Reference Manual. 2025, 10.1.2025. Dostupné z: https://docs.blender.org/manual/en/latest/getting_started/about/. [cit. 2025-01-13].

[12] BROWN, Zack. *A Git Origin Story*. Online. LINUX journal. 2018. Dostupné z: https://www.linuxjournal.com/content/git-origin-story. [cit. 2025-01-14].

[13] W3 SCHOOLS. *XML Introduction*. Online. W3 SCHOOLS. W3 Schools. 1999. Dostupné z: https://www.w3schools.com/xml/xml_whatis.asp. [cit. 2025-01-09].

[14] MEARLS, Mike a CRAWFORD, Jeremy. *Prirucka Hrace*. 10. Vydání. Wizards of the Coast, 2018. ISBN 978-0-7869-6560-1.

[15] PAIZO. *Gamemastering*. Online. D20PFSRD. 2009. Dostupné z: https://www.d20pfsrd.com/gamemastering. [cit. 2025-01-16].

[16] WILLIAMS, Issac. *10 TTRPGs That Define The Genre*. Online. CBR. 1995, s. 1. Dostupné z: https://www.cbr.com/iconic-genre-defining-ttrpgs/. [cit. 2025-01-15].

Seznam použitých symbolů a zkratek

Symbol	Vysvětlení
DnD	Dungeons & Dragons
TTRPG	Table Top Role-Playing Games
FOV	Field of View
PJ	Pán Jeskyně

Seznam obrázků

Seznam výpisů