Střední průmyslová škola a Vyšší odborná škola, Liberec, příspěvková organizace

2D SKÁKAČKA – FORBIDDEN GEM

Maturitní práce

Autor	**Jan Seidel**
Obor	**Informační technologie**
Vedoucí práce	**Ing. Marek Pospíchal**
Školní rok	**2023/2024**
Počet stran	**29**
Počet slov	**3975**

Anotace

Práce pojednává o kompletním vytvoření 2D skákačky vytvořenou v herním enginu Unity. Jejím zaměřením je vyšší obtížnost úrovní se zajímavým příběhem a vlastní pixel art grafikou. Tato práce bere inspiraci z mé oblíbené hry Super Meat Boy.

Summary

This thesis is about the complete creation of a 2D platforming game created in the Unity game engine. Its focus is on higher difficulty levels with an interesting story and custom pixel art graphics. This work takes inspiration from my favourite game Super Meat Boy.

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní práci vypracoval sám a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne 12.03.2024 .....

Jan Seidel

Obsah

Úvod ... 1
1 Použité technologie ... 2
1.1 Vývojové prostředí a programovací jazyk ... 2
1.1.1 Visual Studio ... 2
1.1.2 C# ... 2
1.2 Miro ... 2
1.3 GitHub ... 2
1.4 Unity Engine ... 3
1.5 Aseprite ... 3
2 Unity Engine ... 4
2.1 Prostředí ... 4
2.2 Pojmy ... 5
2.2.1 Game Object ... 5
2.2.2 Asset ... 5
2.2.3 Komponenta ... 5
2.2.4 Transform ... 5
2.2.5 Collider ... 6
2.2.6 Sprite Renderer ... 6
2.2.7 Rigidbody 2D ... 6
2.2.8 Prefab ... 7
2.2.9 Skriptovatelný objekt ... 7
2.2.10 Trigger ... 7
3 Aseprite ... 8
3.1 Prostředí ... 8
3.2 Funkce ... 8
3.2.1 Zobrazení snímků a vrstev ... 8
3.2.2 Onion Skin ... 9
3.2.3 Tilemapy ... 10
3.2.4 Exportování ... 10
4 Vývoj ... 11
4.1 Design ... 11
4.1.1 Vymýšlení příběhu ... 11

4.1.2 Návrh úrovní ... 11
4.1.3 Tvorba grafiky ... 14
4.1.4 Animace ... 15
4.2 Skripty ... 15
4.2.1 Pohyb hráče ... 16
4.2.2 Umírání ... 17
4.2.3 Následování hráče ... 18
4.2.4 Otáčení gravitace ... 18
4.3 Zvuky a hudba ... 19
5 Testování ... 20
5.1 Způsob ... 20
5.2 Výskyty ... 20
5.3 Řešení ... 20
Závěr ... 21
Seznam zkratek a odborných výrazů ... 22
Seznam obrázků ... 23
Použité zdroje ... 24
A. Seznam přiložených souborů ... I

Úvod

Videohry hraji již od velmi brzkého věku se svým otcem. Nejčastěji to byly různé střílečky z první osoby, jako například Call of Duty nebo Payday 2, dále to byly příběhové hry jako série Uncharted, až po skákačky, jako je Super Meat Boy, a právě díky této hře se mi podnítila myšlenka k vytvoření vlastní hry. Dalším z důvodů je, že už dlouho mě nezaujala žádná nová skákačka.

Většinu svých programátorských zkušeností jsem nasbíral na střední škole. Na základní škole jsme v hodinách informačních technologií zkoušeli pouze úplné základy HTML. Na střední škole jsme se učili od vytváření webových stránek a jejich stylování, základy algoritmizace v Pythonu, desktopové a mobilní aplikace v C#, až po umělou inteligenci. Vždy mě ale nejvíce bavilo hrát hry, a proto jsem byl během výuky programování ve čtvrtém ročníku během bloku výuky v herním enginu Unity velice rád za tuto příležitost a motivaci si naprogramovat vlastní hru.

1 Použité technologie

1.1 Vývojové prostředí a programovací jazyk

1.1.1 Visual Studio

Integrované vývojové prostředí (IDE) Visual Studio je kreativní spouštěcí panel, který můžete použít k úpravám, ladění a sestavování kódu a k publikování aplikací. Nad rámec standardního editoru a ladicího programu, které poskytuje většina integrovaných vývojových prostředí (IDE), obsahuje Visual Studio kompilátory, nástroje pro doplňování kódu, grafické návrháře a mnoho dalších funkcí, které zlepšují proces vývoje softwaru.

1.1.2 C#

C# je moderní, objektově orientovaný a typově bezpečný programovací jazyk. Jazyk C# umožňuje vývojářům vytvářet mnoho typů zabezpečených a robustních aplikací, které běží v .NET. Jazyk C# má své kořeny v rodině jazyků C a programátorům v jazyce C, C++, Javě a JavaScriptu je hned povědomý. C# je objektově orientovaný programovací jazyk orientovaný na komponenty. Jazyk C# poskytuje jazykové konstrukce pro přímou podporu těchto konceptů, díky kterým je jazyk C# přirozeným jazykem pro vytváření a používání softwarových komponent. Od svého původu přidává jazyk C# funkce pro podporu nových úloh a nově vznikajících postupů návrhu softwaru. V jádru je jazyk C# objektově orientovaný.

1.2 Miro

Miro je cloudová aplikace pro týmy nebo jednotlivce, kteří chtějí mít na dálku „velkou tabuli společných nápadů“, jako je běžné ve firmách. Uživatelům umožňuje využit velkou plochu pro vkládání obrázků, rozdělovat jí do sekcí pro lepší organizaci, tvořit digitální poznámkové bloky, kreslit a další. Miro je efektivnější způsob organizace práce. Aplikace je dostupná na webu a pro stolní počítače. Miro jsem používal ze začátku práce pro zorganizování svých nápadů. Bylo to velice užitečné.

1.3 GitHub

GitHub je webová služba podporující vývoj softwaru pomocí verzovacího nástroje Git. GitHub nabízí bezplatný webhosting pro open source projekty. GitHub je velmi oblíbený mezi vývojáři po celém světě a je považován za jednu z největších služeb pro správu verzí kódu a spolupráci na projektech.

1.4 Unity Engine

Unity Engine je herní engine. Herní engine je vývojové prostředí určené pro tvorbu videoher pro různé platformy, například počítače, herní konzole, mobilní zařízení a další. Unity Engine obsahuje řadu funkcí, které zrychlují a zjednodušují vývoj nových videoher. Poskytuje vykreslování předmětů nebo herní fyziku. Dále často obsahuje knihovny s připravenými assety, jako jsou různé předměty, modely postav nebo prostředí. To zajišťuje vývojářům příjemnější tvorbu herních světů a prototypů nápadů.

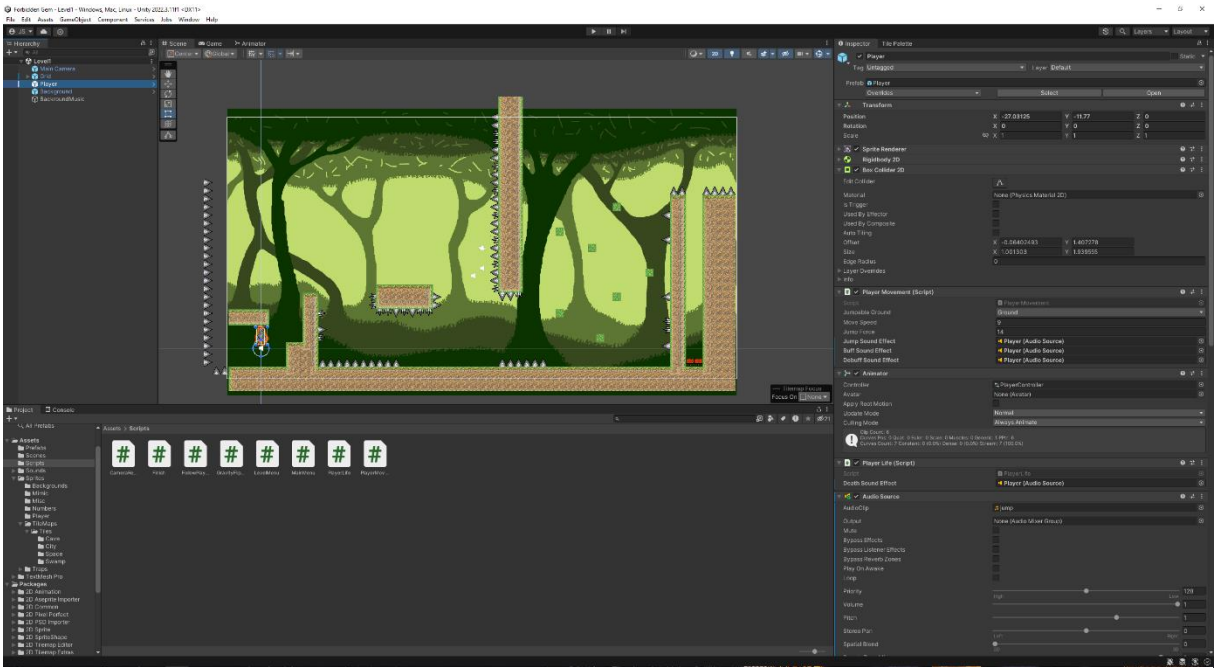
Dalšími herními enginy mimo Unity Engine jsou například Unreal Engine nebo Godot, ale některá herní studia používají vlastní herní engine, jako například Activision Blizzard u Call of Duty. Unity Engine jsem si vybral, protože má dobře vytvořenou dokumentaci a mám zkušenosti z jednoho menšího projektu.

1.5 Aseprite

Aseprite je program pro tvorbu pixel-artové grafiky a animací. Je to placený program, který stojí 19.99$. V žebříčcích nejlepších programů pro tvorbu pixel-artové grafiky se umisťuje mezi nejlepší. Sprity jsou tvořeny vrstvami a framy jako oddělené koncepty. Podporuje velké množství barevných profilů a různých barevných režimů, které jsou RGBA, Indexované (barevné palety pro až 256 barev) a stupně šedi. Jednou z výhod tohoto programu je funkce onion skinning, což znamená, že lze zapnout režim, ve kterém lze pozorovat předchozí snímek, a díky tomu je snazší tvorba plynulých animací. Animace nebo sprity lze buď ukládat jako .aseprite soubory, ve kterých je uchována časová osa animací, nebo jako .png soubory.

2 Unity Engine

2.1 Prostředí



Obrázek 1 Prostředí Unity

Na obrázku výše lze vidět prostředí Unity. Skládá se z podoken, která lze přizpůsobit vlastním potřebám. Na pravé straně se nachází záložka Inspector, která ukazuje podrobnosti zvolených herních objektů. Například právě teď zobrazuje prvky herního objektu Main Camera: Transform, který určuje polohu, natočení a velikost, Camera, která dělá záznam a zobrazuje herní svět hráčům, a Audio Listener, který v podstatě funguje jako mikrofon, protože zaznamenává zvuky vydávané herními prostředky a přehrává je do skutečných reproduktorů. V horní části uprostřed jsou zobrazeny záložky Scene a Game. Tyto záložky přepínají zobrazování mezi hráčským a editorským pohledem. V dolní části uprostřed jsou záložky Project a Console. Záložka Console se používá pro testování možných funkcí, výpisy chyb a výpisy varováni. V záložce Project jsou zobrazeny všechny používané soubory a celé jejich struktury. Na levé straně obrazovky se nachází záložka Hierarchy, ve které se zobrazují všechny herní objekty nacházející se ve stávající scéně.

2.2 Pojmy

2.2.1 Game Object

Herní objekt je základním prvkem Unity enginu reprezentujícím vše, co se nachází v herním světě. Například postavy, speciální efekty, zvuky, lokace nebo nepřátelé. Samostatný herní objekt však nemá žádné vlastnosti ani funkce. K tomu existují komponenty. Každý herní objekt musí mít komponentu transform.

2.2.2 Asset

Asset je reprezentací jakéhokoliv předmětu použitého ve hře nebo projektu. Asset může pocházet ze souboru, který nebyl vytvořen pomocí Unity, jako například 3D model, audio nahrávka, obrázek nebo další typy souborů. Assety vytvořené pomocí Unity mohou být například Animator Controller, který mění spuštěné animace podle aktuální situace, nebo Render Texture, který může být nedílným nástrojem při ladění efektů, které renderují textury.

2.2.3 Komponenta

Komponenty dávají funkce herním objektům. Většina komponentů mívá své vlastní atributy, které určují funkčnost jednotlivých komponent. Jednou z nejdůležitějších komponent v Unity je skript. Skript dokáže definovat vlastní komponenty, ale také zprostředkovávat a řídit cizí komponenty.

2.2.4 Transform

Není jediný herní objekt, který by neměl komponentu Transform. Je to komponenta manipulující a ukládající pozici, velikost a natočení herního objektu. Všechny komponenty Transform mohou mít vlastního rodiče, díky kterému mohou aplikovat pozici, velikost a natočení hierarchicky. To je hierarchie, kterou je možné vidět v záložce Hierarchy. Nadále také podporuje enumerátory a v důsledku jejich použití lze jednoduše procházet jeho potomky.

Obrázek 2 Ukázka Transform komponenty

2.2.5 Collider

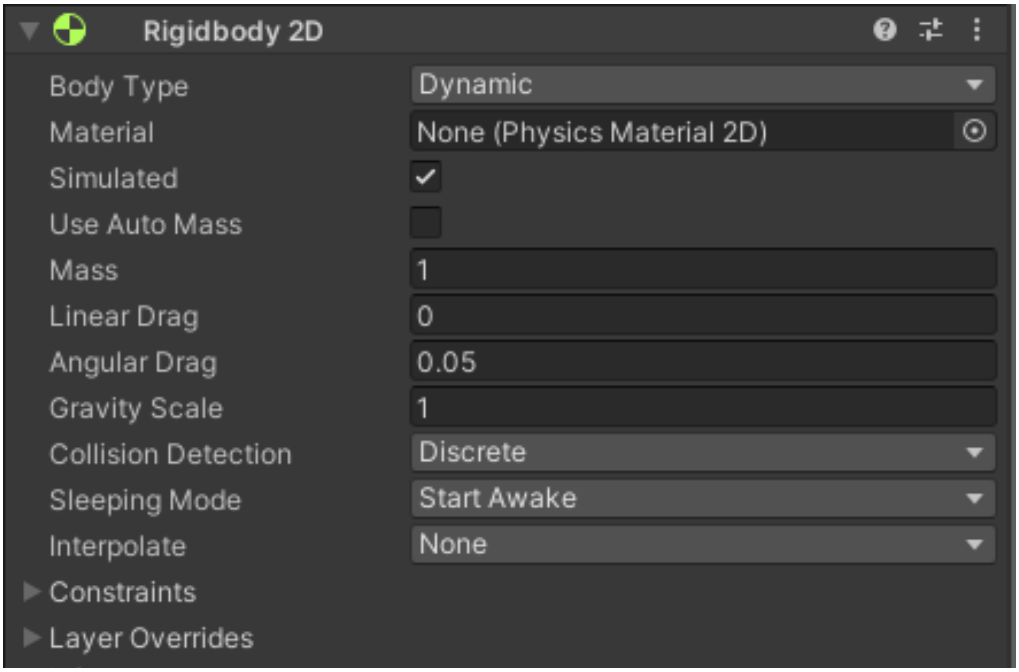
Collider je komponenta sloužící k proměně ve své podstatě objektu na překážku. Do této překážky lze narážet a pokud je potřeba s touto překážkou hýbat, je nutné přidat další komponentu s názvem Rigidbody. Díky důsledkům této komponenty je aplikování fyzikálních vlastností a ovlivnění gravitační silou. Různé skripty na objektu mohou využívat metody „OnCollisionEnter()“, která se vyvolá v případě doteku dalšího collideru nebo rigidbody.

2.2.6 Sprite Renderer

Následující komponenta Sprite Renderer se používá k vykreslování objektu na obrazovku. Komponenta může být deaktivována pro zneviditelnění objektu. Jelikož existuje více typů Rendererů, ale pro mě je jeden z nejdůležitějších Sprite Renderer. Ten se hlavně používá pro vykreslování spritů a k ovládání jejich vizuálního zobrazení na obrazovce.

2.2.7 Rigidbody 2D

Jedna z nejdůležitějších komponent v Unity je Rigidbody 2D. Rigidbody 2D se používá ke spuštění reagování herních objektů na základě fyziky. Jedním z typů reakce je ovlivnění gravitací, následující reakcí může být reakce na další tělesa předáním kinetické energie nebo hybnosti nebo kupříkladu třecí síla.



Obrázek 3 Ukázka Rigidbody 2D komponenty

2.2.8 Prefab

Prefab je systém umožňující vytvoření, nastavení a uložení herních objektů se všemi komponenty, hodnotami atributů a potomky herních objektů. Prefab se používá jako plánek pro používání herního objektu na více místech ve stejné scéně nebo ve více scénách. Díky prefabům je to lehčí než zbytečné kopírování a vkládání identických herních objektů, protože prefab umožňuje mít je všechny synchronizované. To znamená, že když změníme cokoli v prefabu, tak se změna provede ve všech „kopiích“, ale když provedeme změnu v jedné „kopii“, tak se změna stane pouze v ní, a díky tomu lze upravit malé vlastnosti na jednotlivých „kopiích“. Například velikost stromu se může lišit. Také je možnost vnoření prefabů do dalších prefabů, a tím je možné vytvoření komplexní hierarchie objektů, které jsou jednoduché editovat na různých úrovních.

2.2.9 Skriptovatelný objekt

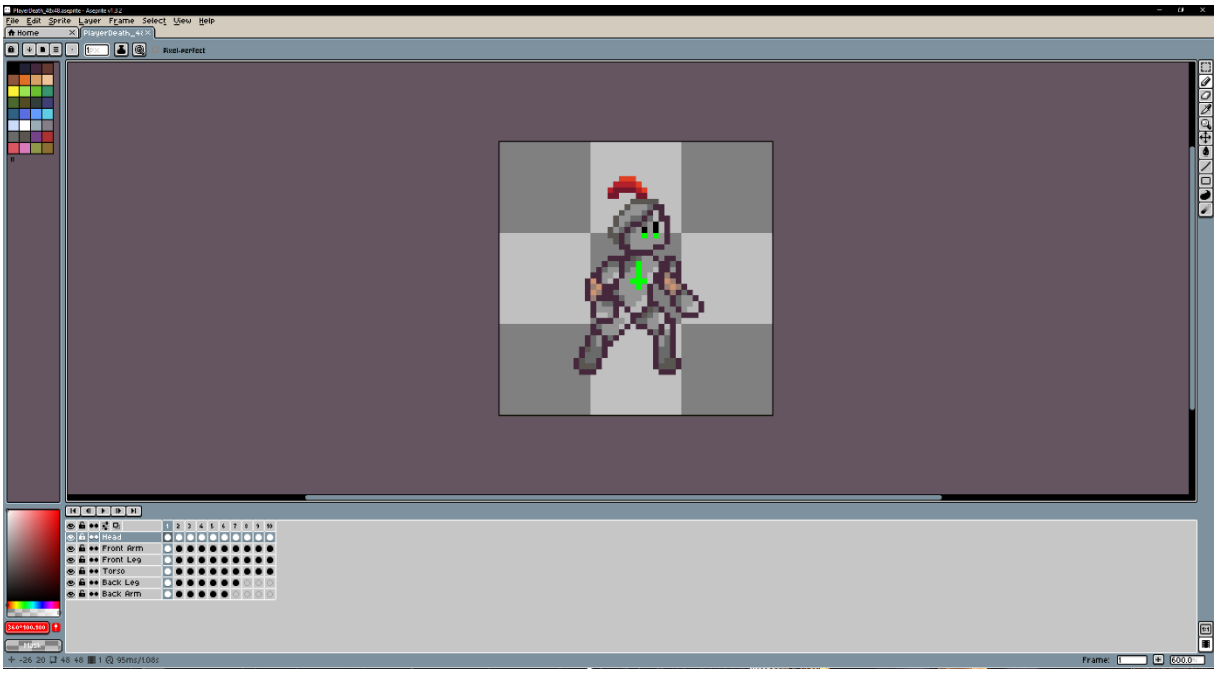
Skriptovatelný objekt je datový prostor s možným použitím pro ukládání velkého množství dat nezávisle od instancí tříd. Jedním z hlavních důvodů použití je snížení používání paměti vašeho projektu, tím že se nepoužívají kopie stejných dat. Je dobré jej používat u projektů, které používají Prefaby ukládající neměnné data v přiložených MonoBehaviour skriptech. Při každé instanci tohoto prefabu vznikne identická kopie dat, čemuž se dá předejít. Na rozdíl od MonoBehaviour skriptů nelze Skriptovatelný objekt připojit k herním objektům a musí být uložen jako Asset v projektu.

2.2.10 Trigger

Trigger je velmi podobný komponentě Collider, protože to je Collider, který neslouží jako překážka, ale jako oblast, při které se volají metody „OnTriggerEnter()“ a „OnTriggerExit()“ pro spuštění skriptů v například důležitých pasážích hry. Výsledkem může být přepnutí hudby během přecházení z jedné lokace do druhé lokace nebo po poražení těžkého nepřítele a dosažení cíle hry.

3 Aseprite

3.1 Prostředí



Obrázek 4 Prostředí Aseprite

Na ukázkovém obrázku výše lze vidět rozhraní této aplikace. Skládá se z několika sekcí, které se dají jednoduše přizpůsobit vlastním potřebám. Na levé straně uprostřed je sekce, která obsahuje barevné palety, a to je výběr mezi základními 65 přednastavenými paletami, jako je například A64, ZX Spectrum, NES, Game Boy a další s možností si vytvořit vlastní paletu. Dále hned pod touto sekcí se nachází výběr barev pomocí RGBA, HSVA, HSLA nebo Gray jako odstín šedé barvy. Hned pod touto sekcí jsou dvě pole, a to barva, která je v popředí a barva, která je v pozadí. Následně se uprostřed obrázku nachází sekce, ve které je kreslící plocha, kterou lze upravit podle vlastních potřeb. Poté ve spodní sekci uprostřed se nachází, kde jsou jednotlivé snímky animace ve sloupci a jednotlivé vrstvy v řádcích. Na pravé straně je sekce, kde se vybírají funkce, jako je například výběr části, tužka, vymazání, výběr barvy z již nakresleného obrázku a další.

3.2 Funkce

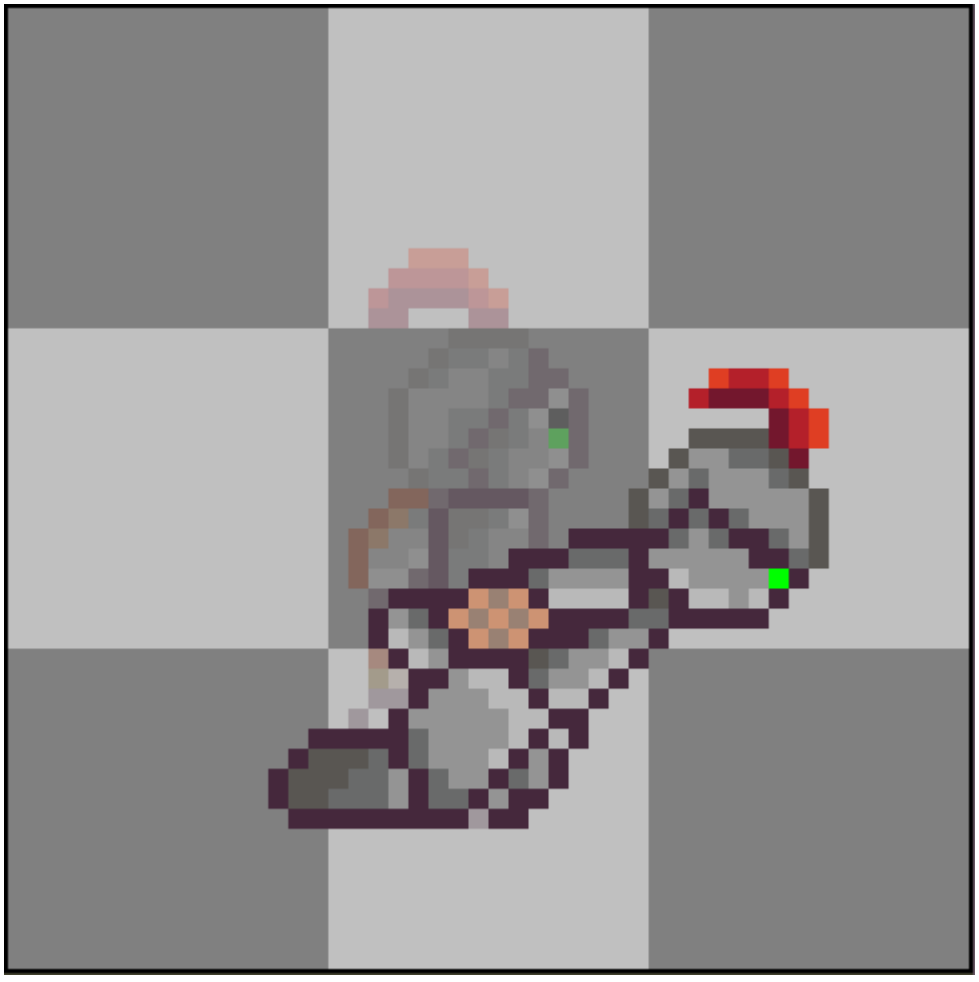
3.2.1 Zobrazení snímků a vrstev

Na výše uvedeném obrázku je ve spodní části sekce zobrazující vrstvy a jednotlivé snímky. Snímky, ale i vrstvy se dají kopírovat, a to se u snímků hodí, aby vznikla plynulejší animace, nebo aby animace byla trochu delší. Jednotlivé vrstvy se dají vypínat. Například

při tvoření postav je výhodné si rozdělit postavu na jednotlivé končetiny pro rychlejší tvorbu snímků. Může se stát, že například se v daném pohybu nebude pohybovat levá ruka, a tak je jednodušší překopírovat pouze levou ruku. U jednotlivých snímků lze nastavit jejich délku v milisekundách.

3.2.2 Onion Skin

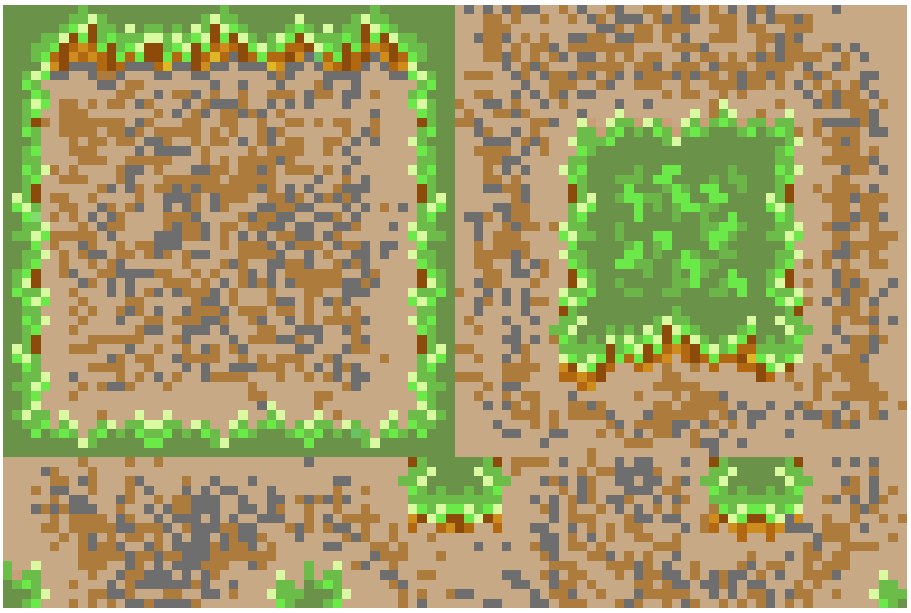
Onion skin je funkce, která umožňuje si zobrazit předcházející snímek pro přesnější kreslení snímku dalšího. Lze si vybrat libovolnou velikost počtu snímků, takže je možné například pozorovat 5 snímků zpět. Tato funkce je jednou z nejlepších v tomto programu a z velké části to je můj důvod výběru této aplikace.



Obrázek 5 Ukázka funkce Onion Skin

3.2.3 Tilemapy

Tilemapy jsou speciální druh vrstev, kde každá buňka plochy na kreslící ploše je referencí na tile v tilesetu. Tile je malý obrázek, který může být použitý na více místech

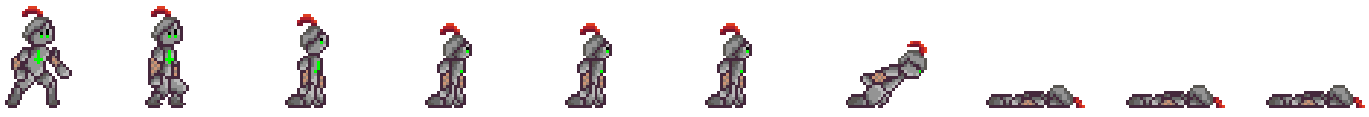


Obrázek 6 Ukázka Tilesetu země v lesní části hry.

v Tilemap vrstvě. Tileset je kolekce tilů o stejné velikosti. Tilemap vrstva je 2D obrázek, kde každý pixel má index pole, který je referencí na určitý tile v tilesetu.

3.2.4 Exportování

Exportování je nedílnou součástí tvorby pixel artu, protože k čemu je vytvořený pixel art, který nemůžeme nikde jinde použít. Animace se dají exportovat jako .gif soubor nebo jako sprite sheet. Lze exportovat třeba jen určitou část kreslící plochy, anebo jen jednotlivé vrstvy nebo snímky animace. Sprite sheet je typ exportu, kde se jednotlivé snímky animace dají za sebe do dlouhé řady a tím vznikne .png soubor.



Obrázek 7 Ukázka Sprite Sheetu animace smrti

V samotném Unity existuje balíček, který umožňuje export rovnou .aseprite souborů a tím urychluje nahrávání animací, protože místo tvoření animace z několika snímků a nastavování časové osy to ten balíček udělá za vás podle nastavení v Aseprite.

4 Vývoj

4.1 Design

4.1.1 Vymýšlení příběhu

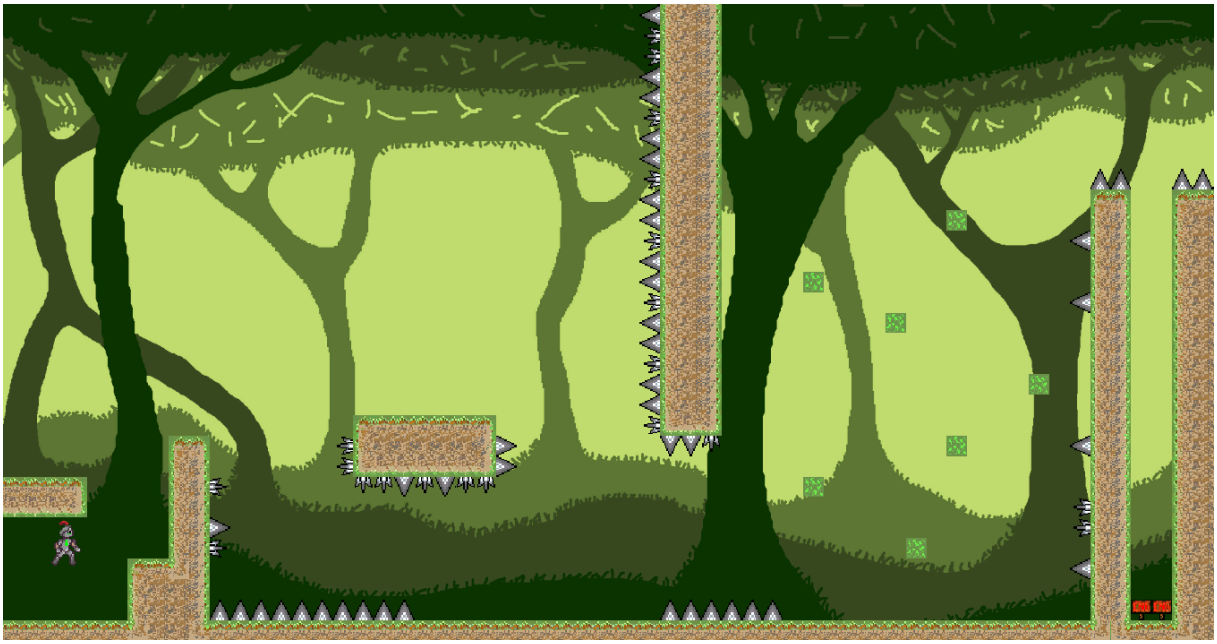
Jako milovník žánru fantasy jsem vždy chtěl vytvořit si vlastní příběh, který bude reprezentovat mou představivost. Proto jsem se rozhodl o vytvoření středověkého světa s nadpřirozenými prvky a na konci s manipulací časoprostorem. Cílem příběhu hry je snaha získání vlastní svobody a následný závěr je důsledkem chamtivosti a touhy po něčem větším. Scénář ke hře je přiložen jako soubor. Hlavní postava Sir Aric je zobrazována jako šlechetný, avšak nemrtvý rytíř, který má silný smysl pro věc. Je odhodlaný splnit svůj úkol a je mu jedno, co ho to bude stát. Morgana, která vyvolala Sira Arica, je tajuplná a velice rozzuřená čarodějnice, která tráví svůj život v temném lesním doupěti plánováním toho, jak sebrat nesmírně silný krystal od krále Rolanda. Král Roland, královský, ale neklidný panovník, je autoritou s historií vyrytou do vrásek jeho znavené tváře.

4.1.2 Návrh úrovní

Návrh úrovní je jednou z nejdůležitějších věcí při vývoji her. Bez zábavných úrovní nikdy nevznikne dobrá hra. Všechny levely jsou lokalizovány podle příběhu hry, který je umístěn ve vedlejším souboru. U každé úrovně jsem se pokusil o jiný styl pixel artu na pozadí.

4.1.2.1 Level 1

Level jedna je základní level, který má ukázat základní funkci skákání. Jeho základní lokace je to umístěna v lese.



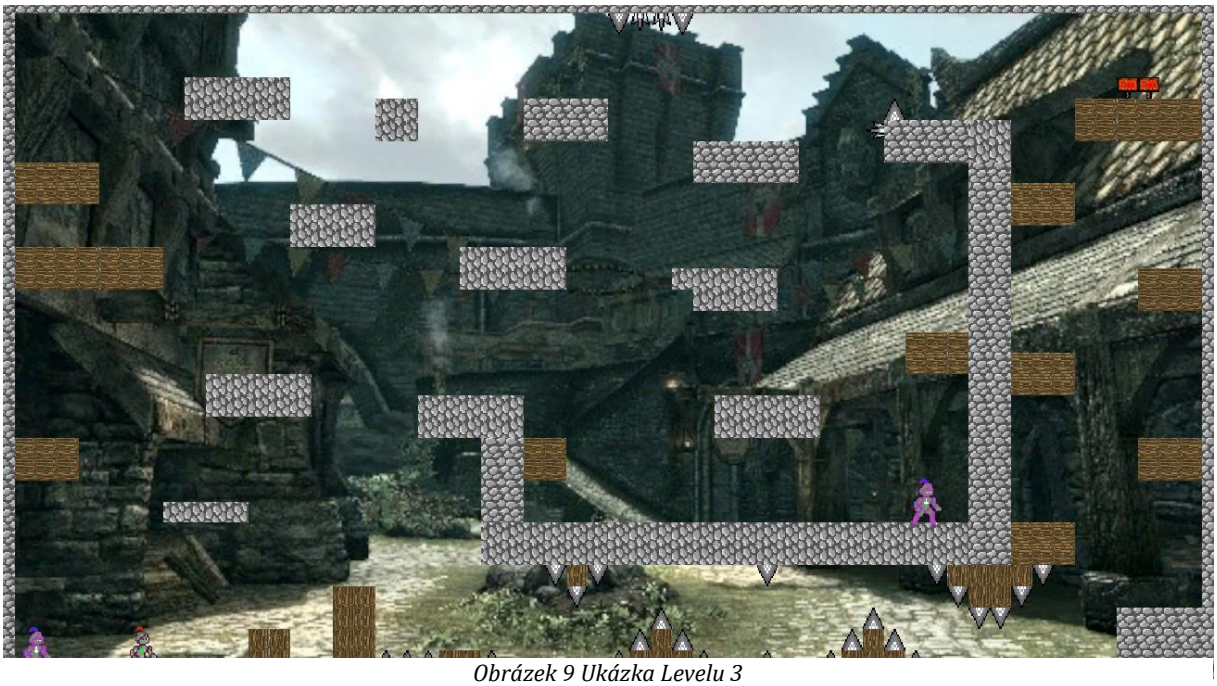
Obrázek 8 Ukázka Levelu 1

4.1.2.2 Level 2

Level dva přivádí novou mechaniku, a to je zrychlování a zpomalování hráče, což ovlivňuje vzdálenost skoků, a to je představeno v této úrovni. Lokalizace tohoto levelu je v bažině.

4.1.2.3 Level 3

Level tři ukazuje mechaniku, kde hráče pronásledují dva mimici. Tato úroveň se nachází ve městě. Při tvorbě pozadí u této úrovně jsem se velice inspiroval městem Solitude ze hry The Elder Scrolls V: Skyrim



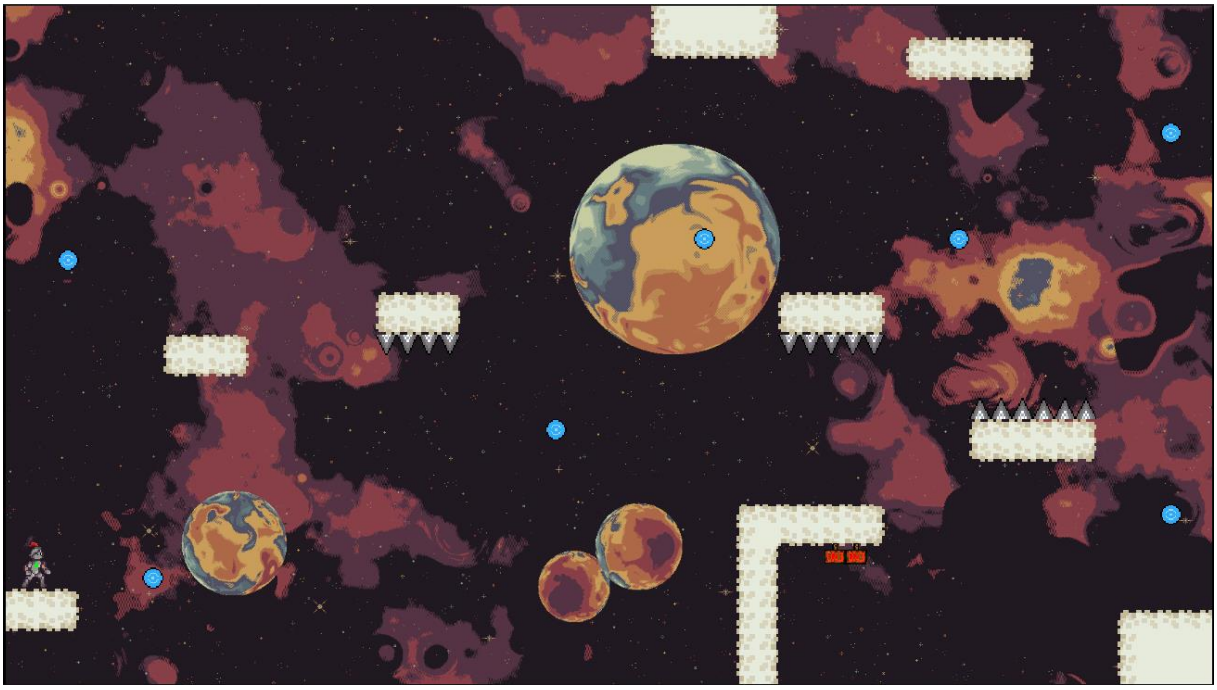
Obrázek 9 Ukázka Levelu 3

4.1.2.4 Level 4

Level čtyři se nachází v jeskyni, a to přináší novou funkcionalitu, a to je zatemněná mapa. Hráč vidí pouze na krátkou vzdálenost kolem sebe. To je vytvořeno překrýváním masek. Jedna je velká černá plocha přes celou mapu a druhá je kruhová plocha následující hráče, která maskuje tu černou plochu a tam se vytvoří viditelná plocha. Assety pro tuto úroveň byli velice složité, a proto jsem je koupil z itch.io od umělce Kauzz za 1$.

4.1.2.5 Level 5

Level 5 je umístěn ve vesmírné lokalitě, kde se skáče po asteroidech. Hlavní mechanika tohoto levelu je otáčení gravitace pomocí modrých orbů.



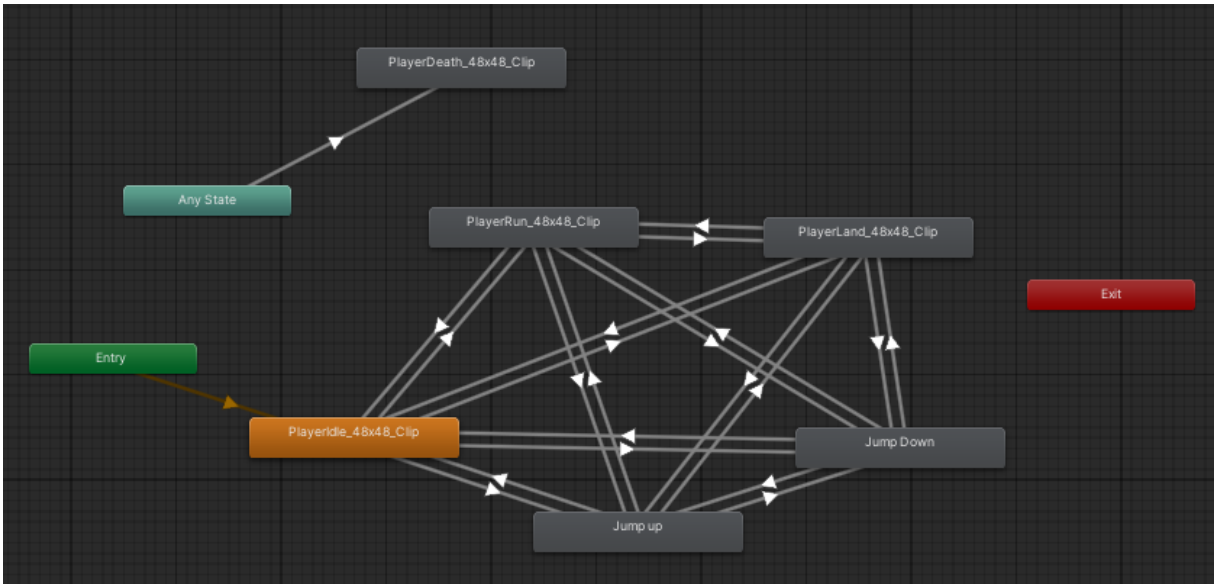
Obrázek 10 Ukázka Levelu 5

4.1.3 Tvorba grafiky

Jako někdo, kdo v podstatě nikdy netvořil grafiku, jsem se rozhodl, že si to chci vyzkoušet a nasbírat zkušenosti. Proto mě napadlo, že když budu muset vytvořit celý pixel art styl pro vlastní hru, tak nebudu moci použít existující assety a vše si budu muset vytvořit sám. To je ohromná motivace se tomu začít věnovat. Pro pixel art styl jsem se rozhodl, protože pixelová mřížka poskytuje strukturovaný systém, který je pro mě jako nováčka v grafickém designu snadněji pochopitelný. Dalším důvodem výběru pixel art stylu je retro estetický vzhled. Jednoduchost pixel artu může evokovat pocit klasických videoher a poskytnout hře jedinečný a vizuálně působivý vzhled.

4.1.4 Animace

Když už se vytváří pixel art grafika u hry, tak je samozřejmostí, že bude potřeba také tvořit vlastní animace. Samotné nízké rozlišení pixel artu zjednodušuje proces tvoření animací a usnadňuje ho začátečníkům, jako jsem já, pro které by tradiční animační techniky mohly být příliš obtížné. Pro samotnou implementaci do Unity se v Unity používá animator. V něm se určuje návaznost animací, a to určuji podle prvku state, který se nastavuje podle aktuálního pohybu hráče.



Obrázek 11 Ukázka animatoru

4.2 Skripty

V Unity jsou skripty malé programy napsané v programovacím jazyce C#. Jejich funkcí je definování chování a funkcionality ke Game Objektům. K jejich vytváření je dobře přizpůsobené vývojové prostředí Microsoft Visual Studio 2022.

4.2.1 Pohyb hráče

Pohyb hráče je velice důležitý a bez pohybu by většina her neexistovala. Já k pohybovým tlačítkům používám Input Manager, kde pohyb do stran je nastavený na Horizontal, kde to je nastavené na šipky nebo písmena A a D. Na skákání používám Jump, kde je to nastavené na tlačítka mezerník a W. Ke skákání bylo potřeba implementovat funkci, která kontroluje, zda je hráč uzemněný, a to proto, aby hráč nemohl skákat, když je ve vzduchu.

```
Zpráva Unity | Počet odkazů: 0
35  private void Update()
36  {
37      dirX = Input.GetAxisRaw("Horizontal");
38      rb.velocity = new Vector2(dirX * moveSpeed, rb.velocity.y);
39
40      if (Input.GetButtonDown("Cancel"))
41      {
42          SceneManager.LoadScene("Menu");
43      }
44
45      if (Input.GetButtonDown("Jump") && IsGrounded())
46      {
47          rb.velocity = new Vector2(rb.velocity.x, jumpForce);
48      }
49
50      UpdateAnimationState();
51  }
```

Obrázek 12 Ukázka kódu pohybu

4.2.2 Umírání

Situace úmrtí nastane, když se hráč dotkne objektu, který má tag s názvem Trap. To spustí funkci Die(), která akorát vypne ovládání a spustí animaci smrti. Po ukončení samotné animace je potom odkaz na funkci RestartLevel(), která znovu načte ten stejný level.

```
Skript Unity (počet odkazů na prostředky: 1) | Počet odkazů: 0
public class PlayerLife : MonoBehaviour
{
    private Animator anim;
    private PlayerMovement pm;
    private bool isDead = false;

    Zpráva Unity | Počet odkazů: 0
    private void Start()
    {
```

Obrázek 13 Ukázka kódu pro skákání

```
        pm = GetComponent<PlayerMovement>();
    }

    Zpráva Unity | Počet odkazů: 0
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (!isDead && collision.gameObject.CompareTag("Trap"))
        {
            Die();
        }
    }

    Počet odkazů: 1
    private void Die()
    {
        isDead = true;
        pm.enabled = false;
        anim.SetTrigger("death");
    }

    Počet odkazů: 0
    private void RestartLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```

Obrázek 14 Ukázka kódu smrti

4.2.3 Následování hráče

Následování hráče je skoro stejné jako pohyb hráče. V Unity mám vytvořený objekt, který bude následovat hráče. K němu je připojený transform (pozice) samotného hráče. To potom používám k lokalizaci hráče a toho potom následuji.

```
Zpráva Unity | Počet odkazů: 0
27  private void Update()
28  {
29      Vector2 direction = (playerTransform.position - transform.position).normalized;
30      rb.velocity = new Vector2(direction.x * moveSpeed, rb.velocity.y);
31
32      bool isGrounded = IsGrounded();
33
34      bool canReachPlayer = CanReachPlayer();
35
36      if (isGrounded && !canReachPlayer && !canJump)
37      {
38          rb.velocity = new Vector2(rb.velocity.x, jumpForce);
39          canJump = true;
40      }
41      UpdateAnimationState();
42  }
```

Obrázek 15 Ukázka kódu následování hráče

4.2.4 Otáčení gravitace

Otáčení gravitace je uděláno velice jednoduše, a to tím, že se ve Physics2D atribut gravity vynásobí -1, a tím se gravitace otočí.

```
13  private void OnTriggerEnter2D(Collider2D collision)
14  {
15      if (collision.gameObject.name == "Player")
16      {
17          Physics2D.gravity *= -1;
18          gravitySoundEffect.Play();
19          collision.transform.Rotate(Vector3.forward, 180f);
20      }
21  }
```

Obrázek 16 Ukázka kódu otáčení gravitace

To otočí gravitaci, ale neotočí to samotný model hráče. To jsem musel udělat pouze rotací modelu. To dobře otočilo model hráče, ale animace se přehrávali naopak. To jsem vyřešil kontrolou, zda je gravitace otočená, anebo ne, při updatování animací a potom animaci otočit o 180 stupňů. Poté se objevil další problém, který znemožňoval skákat.

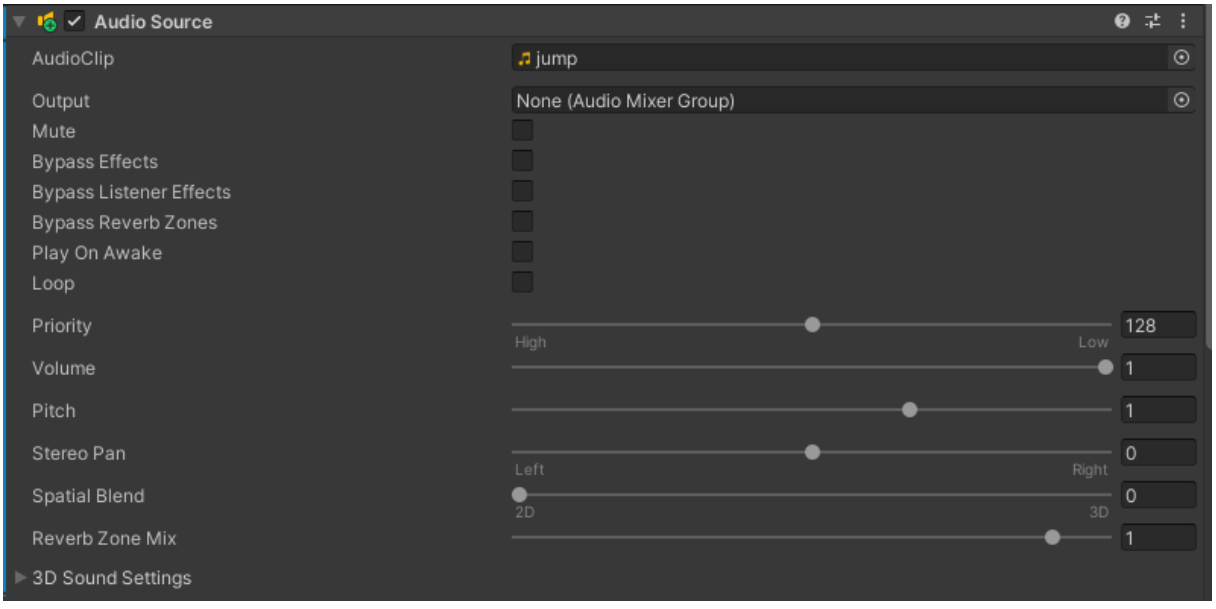
Problém nastal při kontrole, zda je hráč uzemněn. Dříve jsem kontroloval, zda je zem jen pod hráčem, a jelikož byl teď hráč otočen, tak to kontrolovalo plochu ve vzduchu. To jsem vyřešil kontrolou gravitace a vykreslování kontrolujícího obrazce na správnou stranu.

```
116     private bool IsGrounded()
117     {
118         float gravityDirection = Mathf.Sign(Physics2D.gravity.y);
119         if(gravityDirection > 0f)
120         {
121             return Physics2D.BoxCast(coll.bounds.center, coll.bounds.size, 0f, Vector2.up, .1f, jumpableGround);
122         }
123         else
124         {
125             return Physics2D.BoxCast(coll.bounds.center, coll.bounds.size, 0f, Vector2.down, .1f, jumpableGround);
126         }
127 
128     }
```

Obrázek 17 Ukázka upraveného kódu uzemnění

4.3 Zvuky a hudba

Aby ve hře byly slyšet zvuky, jsou potřeba dvě části, Audio Listener a Audio Source. Audio Listener je komponenta, která je pouze posluchač. Ze začátku je umístěna na Main Camera. Poté Audio Source je komponenta na objektu, který ten daný zvuk bude vydávat. To se pak používá přímo ve skriptech mechanik pro přehrání zvuku po akci. Jediný rozdíl mezi zvuky hry a hudbou je ten, že u komponenty Audio Source nechávám zapnutou možnost loop pro přehrání hudby, i když hudba skončí. Audio jsem si chtěl udělat vlastní, ale nebyl na to čas, a proto jsem použil zvuky z Unity Asset Storu.



Obrázek 18 Ukázka komponenty Audio Source

5 Testování

5.1 Způsob

Testování jsem prováděl po každé nové přidané funkci. Vždy jsem se pokusil objevit nejrůznější způsoby, jak hru rozbít a občas se to i povedlo. Potom co jsem opravil všechny chyby, co jsem našel, tak jsem to dal zahrát kamarádovy, aby jako on s novým pohledem zkusil přijít na nový způsob, jak hru rozbít.

5.2 Výskyty

Pár chyb po implementaci nových funkcí se objevilo. Například při první implementaci skákání se mohlo skákat neomezeně, nebo při implementaci otáčení gravitace v Levelu 5 se špatně otáčela animace.

5.3 Řešení

Pro vyřešení prvního uvedeného příkladu minulé podkapitoly stačilo implementovat funkci pro vykreslování neviditelného čtverce pod model hráče, který když se dotýká země, tak umožňuje spustit metodu pro skákání. Pro druhý příklad jsem implementoval funkci kontrolující světovou gravitaci a podle toho jsem otočil animaci podle osy X.

Závěr

Touto prací jsem zjistil, že vývoj her je velice zábavnou, ale těžkou prací. Vždy jsem věděl, že vývoj her je složitý, ale když jsem si to vyzkoušel sám tak jsem viděl reálný časový náklad, který se vývoji her musí obětovat. Určitě mám v plánu pokračovat na vývoji dalších her. Vývoj této hry mě naučil spoustu nových technologií a postupů. Vyzkoušel jsem si a zjistil jsem, že na vývoji této hry mně nejdéle trvalo dělat grafickou část. Díky tomu jsem se docela dobře naučil v Asepritu a to je pro mě velice cenné. Při vývoji této hry vznikla podle mého usouzení zábavná, ale poměrně obtížná hra a dokument popisující její vývoj. Před vývojem jsem očekával, že nejdelší časovým nákladem bude sestavení samotné hry, ale to nebyla pravda. Sestavení samotné hry mi zabralo přibližně 12 hodin. Do toho počítám vymyšlení rozložení levelů a jejich samotné vytvoření a napsání skriptů. Dále mi přibližně 6 hodin zabralo testování a opravování chyb. A nejdelší časový náklad bylo tvoření vlastních animací a pozadí. Všechny hráčské animace byli moje první zkušenost s animací a zároveň s tvořením pixel art grafiky, a proto mi trvali vytvořit přibližně 7 hodin. Pozadí z levelů 1 až 4 jsem dohromady tvořil asi 17 hodin. Největší práci jsem si dal s pozadím u levelu 5. To je pozadí, které se nachází ve vesmíru. Vytvoření samotné hlavní planety v pozadí mi zabralo 6 hodin. Dohromady mi pozadí z levelu 5 zabralo 21 hodin čistého času. Z pozadí u levelu 5 mám obrovskou radost. Dále se těším, až své nově naučené schopnosti použiji při dalším vývoji her a vývoj mě bude bavit dál.

Seznam zkratek a odborných výrazů

Engine
Je v informatice označení pro ústřední část softwarového produktu, v kontextu videoher je to platforma, která poskytuje nástroje pro tvorbu her

Game Object
Základní stavební prvek herního enginu, reprezentuje vše, co se ve hře nachází

Komponenta
Nastavuje funkcionalitu a chování game objectu

Asset, asset store
Aktivum, místo pro jejich nákup

Prafab
Asset, který lze používat dokola a změna v nadřazeném se provede ve všech

Skript
Kód definující chování objektu

Level
Herní úroveň

Onion skinning
Vrstvení jednotlivých snímků při tvorbě pixel artu

Seznam obrázků

Obrázek 1 Prostředí Unity 4
Obrázek 2 Ukázka Transform komponenty 5
Obrázek 3 Ukázka Rigidbody 2D komponenty 6
Obrázek 4 Prostředí Aseprite 8
Obrázek 5 Ukázka funkce Onion Skin 9
Obrázek 6 Ukázka Tilesetu země v lesní části hry. 10
Obrázek 7 Ukázka Sprite Sheetu animace smrti 10
Obrázek 8 Ukázka Levelu 1 12
Obrázek 9 Ukázka Levelu 3 13
Obrázek 10 Ukázka Levelu 5 14
Obrázek 11 Ukázka animatoru 15
Obrázek 12 Ukázka kódu pohybu 16
Obrázek 13 Ukázka kódu pro skákání 17
Obrázek 14 Ukázka kódu smrti 17
Obrázek 15 Ukázka kódu následování hráče 18
Obrázek 16 Ukázka kódu otáčení gravitace 18
Obrázek 17 Ukázka upraveného kódu uzemnění 19
Obrázek 18 Ukázka komponenty Audio Source 19

Použité zdroje

1. ***Miro.com.*** **[Online] [Citace: 15. 11 2023.] https://miro.com.**

2. Wikipedie, Přispěvatelé. Unity (herní engine). ***Wikipedie: Otevřená encyklopedie.*** **[Online] 12. 11 2023. [Citace: 15. 11 2023.] https://cs.wikipedia.org/w/index.php?title=Unity_(hern%C3%AD_engine)&oldid=23372746.**

3. Igara Studio. Aseprite Docs. ***Aseprite.*** **[Online] 28. 12 2023. [Citace: 28. 12 2023.] https://www.aseprite.org/docs/.**

4. Unity Technologies. Unity Documentation. ***Unity.*** **[Online] 13. 11 2023. [Citace: 15. 11 2023.] https://docs.unity3d.com/Manual/index.html.**

5. GitHub, Inc. Github Docs. ***Github.*** **[Online] 28. 12 2023. [Citace: 28. 12 2023.] https://docs.github.com/en.**

6. Kauzz. Cave Tiles. [Online] 8. 9 2021. [Citace: 11. 3 2024.] https://kauzz.itch.io/kpc.

7. B.G.M. Casual Game BGM #5. [Online] 19. 1 2019. [Citace: 15. 2 2024.] https://assetstore.unity.com/packages/audio/music/casual-game-bgm-5-135943.

8. Dustyroom. FREE Casual Game SFX Pack. [Online] 4. 3 2019. [Citace: 15. 2 2024.] https://assetstore.unity.com/packages/audio/sound-fx/free-casual-game-sfx-pack-54116#description.

A. Seznam přiložených souborů

Na přiloženém datovém nosiči se nacházejí následující soubory a složky:

- **MP2023-2024-P4-Seidel-Jan.docx** – editovatelná verze dokumentace maturitní práce
- **MP2023-2024-P4-Seidel-Jan.pdf** – tisknutelná verze dokumentace maturitní práce
- **Forbidden Gem** – Samotná hra