

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

technická dokumentace

3D hra v Unity

**INFORMAČNÍ
TECHNOLOGIE
18-20-M/01
se zaměřením na vývoj
webových aplikací**

**Marián Kufa
IT4
Střední škola průmyslová
a umělecká, Opava
Školní rok: 2025/26**

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8

V Opavě dne 6.1.2026

podpis

Abstrakt

Tato práce se zabývá vývojem 3D survival hry v herním enginu Unity s využitím programovacího jazyka C#. Hra kombinuje prvky střílečky a survival žánru, kde hráč čelí vlnám zombie nepřátel a snaží se dosáhnout co nejvyššího skóre. Projekt zahrnuje kompletní vývoj od prototypu až po funkční hru s několika scénami, systémem skóre, umělou inteligencí nepřátel využívající NavMesh pro navigaci, systémem životů a výdrže, a uživatelským rozhraním. Hra se stává postupně obtížnější, protože se nepřátelé zrychlují, náhodným respawnem hráče i nepřátel. Grafické prostředí bylo vytvořeno pomocí Unity Terrain Tools a volně dostupných 3D modelů.

Klíčová slova: Unity, C#, game development, survival, zombie, NavMesh, AI, 3D hra

Abstract

This thesis deals with the development of a 3D survival game in the Unity game engine using the C# programming language. The game combines elements of the shooter and survival genres, where the player faces waves of zombie enemies and tries to achieve the highest score possible. The project includes complete development from prototype to a functional game with several scenes, a scoring system, enemy artificial intelligence using NavMesh for navigation, a life and stamina system, and a user interface. The game features a dynamic difficulty system where enemies gradually speed up, random respawning of players and enemies. The graphical environment was created using Unity Terrain Tools and freely available 3D models.

Key words: Unity, C#, game development, survival, zombie, NavMesh, AI, 3D game

Obsah

Abstrakt	3
Úvod	1
1 Teoretická a metodická východiska	2
1.1 Herní enginy a jejich role ve vývoji her	2
1.2 Unity jako vývojové prostředí	2
1.3 Programování v C#	2
1.4 Principy umělé inteligence ve hrách	3
2 Využité technologie	4
2.1 Unity Engine verze 2021.3 LTS	4
2.2 Visual Studio 2022	4
2.3 Unity Asset Store	5
2.4 Unity NavMesh systém	5
2.5 Zvukové efekty z Freesound.org	5
2.6 Unity Terrain Tools	5
3 Způsoby řešení a použité postupy	6
3.1 Implementace pohybového systému	6
3.2 Systém střelby s raycast technologií	7
3.3 Umělá inteligence zombie s NavMesh	8
3.4 Systém skóre a scén	9
3.5 Tvorba herního světa	10
3.6 Uživatelské rozhraní	11
4 Výsledky řešení a výstupy	12
4.1 Splnění cílů práce	12
4.2 Vizuální a audio stránka	12
4.3 Výkon a optimalizace	12
4.4 Testování a ladění	13
4.5 Uživatelský manuál	14
Závěr	15
Seznam použitých informačních zdrojů	16

Úvod

Jako svůj závěrečný projekt jsem si zvolil vytvoření 3D survival hry v herním enginu Unity. Už od mala mě fascinovaly počítačové hry a vždy mě zajímalo, jak vlastně fungují a co vše je potřeba k jejich vytvoření. Rozhodl jsem se proto využít příležitost závěrečné práce k tomu, abych si vyzkoušel celý proces vývoje hry od prvotního nápadu až po finální produkt.

Cílem této práce je vytvořit plně funkční 3D survival hru s následujícími charakteristikami: implementace pohybového systému s pokročilými mechanikami, vytvoření systému střelby využívajícího raycast technologii, návrh a programování umělé inteligence nepřátel pomocí NavMesh, design uživatelského rozhraní zobrazujícího všechny klíčové informace, a implementace systému skóre s persistentním ukládáním dat.

Výsledná hra kombinuje prvky survival žánru se systémem progresivně rostoucí obtížnosti. Hráč se ocitá v otevřeném prostředí, kde musí zombie nepřátelům, přičemž jeho úkolem je přežít co nejdéle a dosáhnout co nejvyšší skóre. Projekt mi umožnil prakticky aplikovat znalosti programování v jazyce C# a zároveň se naučit pracovat s profesionálním herním enginem Unity.

1 Teoretická a metodická východiska

1.1 Herní enginey a jejich role ve vývoji her

Herní engine je softwarový framework navržený pro tvorbu a vývoj počítačových her. Poskytuje vývojářům hotové nástroje a systémy pro práci s grafikou, fyzikou, zvukem, umělou inteligencí a dalšími aspekty herního vývoje. Podle studie Unity Technologies z roku 2024 využívá Unity více než 60% všech mobilních her a je třetím nejpoužívanějším herním enginem na světě.

Hlavní výhodou použití herního enginu oproti programování hry od základů je výrazné zkrácení vývojového času. Engine poskytuje hotové řešení pro složité problémy jako je renderování 3D grafiky, detekce kolizí, správa animací nebo fyzikální simulace. To umožňuje vývojářům soustředit se na samotný herní design a mechaniky namísto řešení low-level technických problémů.

1.2 Unity jako vývojové prostředí

Unity patří mezi nejrozšířenější a nejvíce používané herní enginey na světě. Je vhodný jak pro začátečníky, tak pro pokročilé vývojáře a umožňuje tvorbu her pro široké spektrum platform. Unity vzniklo v roce 2005 a od té doby se stalo standardem v oblasti indie game developmentu i komerčního vývoje.

Jednou z hlavních výhod Unity je jeho multiplatformnost. Engine podporuje export na více než 25 platform včetně Windows, macOS, Linux, iOS, Android, konzolí PlayStation a Xbox, ale i webových prohlížečů. To znamená, že hra vytvořená v Unity může být snadno portována na různá zařízení bez nutnosti přepisování celého kódu.

1.3 Programování v C#

Unity využívá jako primární skriptovací jazyk C#, což je moderní objektově orientovaný jazyk vyvinutý společností Microsoft. C# je staticky typovaný jazyk, což znamená, že typy proměnných musí být definovány při kompilaci. To snižuje počet runtime chyb a usnadňuje údržbu kódu. Podle dokumentace Microsoft Learn je C# navržen pro tvorbu různých aplikací na platformě .NET.

Výhodou C# je jeho čitelnost, výkonnost a podpora moderních programovacích konceptů jako jsou delegáti, lambda výrazy, LINQ dotazy a asynchronní programování. Jazyk také podporuje automatickou správu paměti pomocí garbage collectoru, což zjednodušuje práci s alokací a dealokací paměti.

1.4 Principy umělé inteligence ve hrách

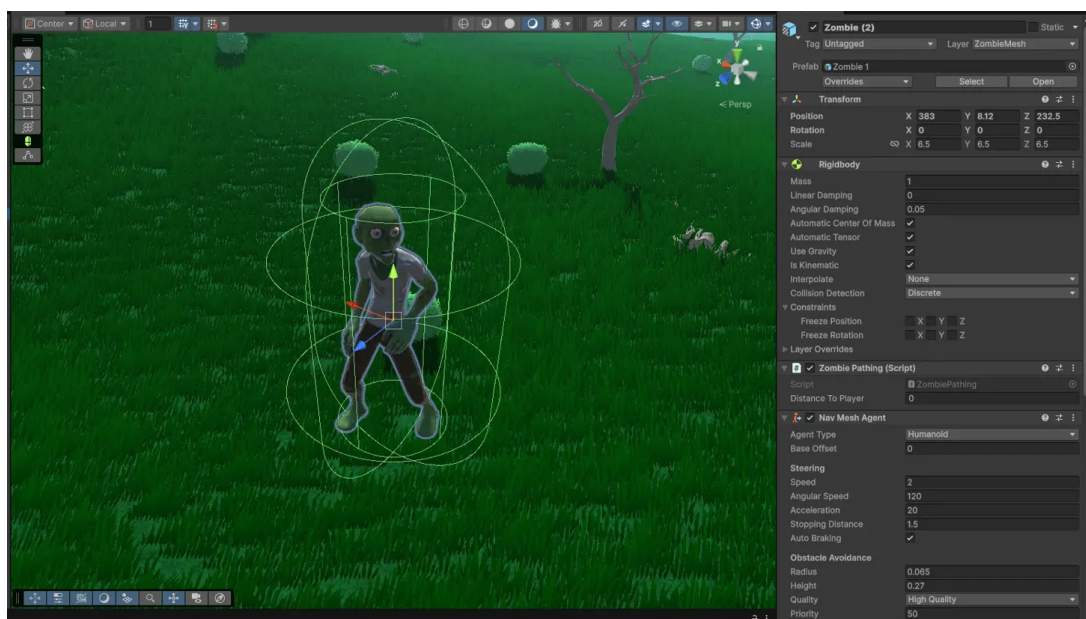
Umělá inteligence ve hrách se označuje jako Game AI a liší se od akademické AI tím, že prioritou není dokonalá inteligence, ale zábavný a předvídatelný herní zážitek. Podle Unity Documentation je NavMesh systém jedním z nejpoužívanějších nástrojů pro implementaci AI navigace.

NavMesh (Navigation Mesh) je reprezentace herního světa jako sítě polygonů, po kterých se mohou AI agenti pohybovat. Systém automaticky vypočítává nejkratší cestu mezi dvěma body a obchází překážky. To umožňuje vytváření inteligentního chování nepřátel bez nutnosti ručního programování pathfindingu.

2 Využité technologie

2.1 Unity Engine verze 2021.3 LTS

Pro vývoj hry jsem zvolil Unity verze 2021.3 LTS (Long Term Support). Tato verze poskytuje stabilní prostředí s dlouhodobou podporou, což je ideální pro studijní projekty. Unity Editor obsahuje všechny potřebné nástroje pro vývoj včetně Scene View pro vizuální úpravu světa, Inspector pro nastavení komponent, Console pro ladění a Game View pro testování hry.



Obrázek 1: Unity Editor s nastaveným zombie objektem

Unity Editor je komplexní vývojové prostředí, které v základním nastavení může působit nepřehledně. Proto je doporučeno přizpůsobit si rozhraní podle vlastních potřeb. Scene View je 3D prostor, kde se vizuálně upravuje herní svět. Hierarchy zobrazuje hierarchický seznam všech objektů přítomných v aktuální scéně. Project panel funguje jako správce projektových souborů a obsahuje všechny assets použité v projektu.

Inspector je panel pro detailní nastavení vybraného objektu, kde se upravují všechny komponenty objektu jako skripty, materiály, collidery a fyzikální vlastnosti. Console je nepostradatelný nástroj pro vývojáře, který zobrazuje tři typy zpráv - Log zprávy pro informace, Warning zprávy pro varování a Error zprávy pro chyby.

2.2 Visual Studio 2022

Jako vývojové prostředí pro psaní C# skriptů jsem používal Visual Studio 2022, které je plně integrováno s Unity. Visual Studio poskytuje IntelliSense pro automatické dokon-

čování kódu, debugging nástroje pro krokování programu a detekci chyb, a integraci s Unity pro rychlé otevírání skriptů.

2.3 Unity Asset Store

Unity Asset Store je obchod s tisíci volně dostupných i placených assetů. Pro svou hru jsem využil několik bezplatných asset balíčků:

Lowpoly Environment - Nature Free obsahuje low-poly modely stromů, keřů, kamenů, hub a tráv. Tento balíček jsem použil pro vytvoření herního prostředí s více než deseti tisíci stromy.

Low Poly Zombie ze Sketchfab je riggovaný model zombie s třemi animacemi - chůze, útok a smrt. Model jsem musel upravit přidáním colliderů pro detekci zásahů.

Pistol z Poly Pizza je stylizovaná low-poly laserová pistole, která zapadá do celkového vizuálního stylu hry.

2.4 Unity NavMesh systém

NavMesh je vestavěný systém Unity pro navigaci AI agentů. Poskytuje automatické hledání cest, vyhýbání se překážkám a optimalizovaný výkon. Při vytváření mapy jsem musel označit všechny statické objekty jako Navigation Static a poté vygenerovat NavMesh pomocí vestavěného nástroje.

2.5 Zvukové efekty z Freesound.org

Pro zvuk laserové zbraně jsem použil efekt "Plasma - Lazer Pistol Gun Shot 1" od uživatele Erokia z platformy Freesound.org, která poskytuje royalty-free zvukové efekty. Zvuk byl ve formátu MP3 a byl importován do Unity, kde byl přiřazen k audio zdroji.

2.6 Unity Terrain Tools

Pro tvorbu herního světa jsem využil Unity Terrain systém, který umožňuje vytváření rozlehlých krajin s kopci, údolími a různými texturami. Terrain Tools poskytují brush nástroje pro modelování terénu, paint nástroje pro aplikaci textur, a mass placement nástroje pro hromadné umísťování vegetace.

3 Způsoby řešení a použité postupy

3.1 Implementace pohybového systému

Pohybový systém hráče byl implementován pomocí komponenty Rigidbody, která zajišťuje realistické fyzikální chování. Rigidbody umožňuje aplikovat síly, upravovat rychlost a detekovat kolize s jinými objekty.

```
// Movement calculation and adjustments
void MovePlayer()
{
    // Direction
    moveDirection = orientation.forward * verticalInput + orientation.right * horizontalInput;

    // Ground
    if (grounded)
    {
        rb.AddForce(moveDirection.normalized * moveSpeed * 10f, ForceMode.Force);
    }

    // Air
    else if (!grounded)
    {
        rb.AddForce(moveDirection.normalized * moveSpeed * 10f * airMultiplier, ForceMode.Force);
    }
}

void SpeedControl()
{
    Vector3 flatVel = new Vector3(rb.linearVelocity.x, 0f, rb.linearVelocity.z);

    // Velocity limiter
    if (flatVel.magnitude > moveSpeed)
    {
        Vector3 limitedVel = flatVel.normalized * moveSpeed;
        rb.linearVelocity = new Vector3(limitedVel.x, rb.linearVelocity.y, limitedVel.z);
    }
}

void Jump()
{
    rb.linearVelocity = new Vector3(rb.linearVelocity.x, 0f, rb.linearVelocity.z);

    rb.AddForce(transform.up * jumpForce, ForceMode.Impulse);
}

void JumpReset()
{
    readyToJump = true;
}
}
```

Obrázek 2: Kód implementující pohybový systém hráče

Pohyb je realizován tak, že se nejprve zachytí vstup z klávesnice pomocí Input.GetAxisRaw funkce pro osy "Vertical" a "Horizontal". Směr pohybu se vypočítá relativně k orientaci hráče pomocí transform.forward a transform.right vektorů. Pokud je hráč na zemi (detekováno pomocí raycastu dolů), aplikuje se plná síla pohybu. Pokud je ve vzduchu, aplikuje se pouze část síly vynásobená air multiplierem (0.4), což simuluje sníženou kontrolu ve vzduchu.

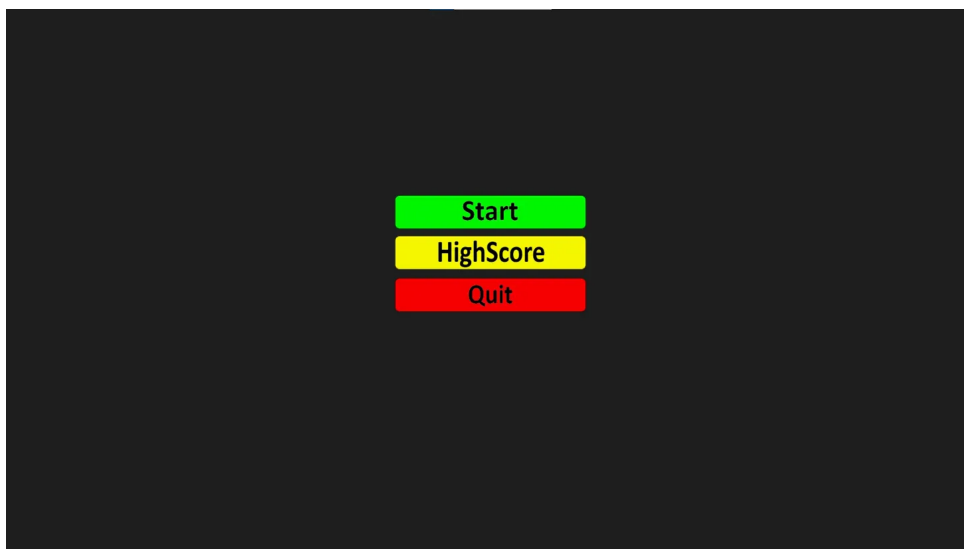
Sprint je implementován jako zvýšení rychlosti při držení klávesy Left Shift. Sprintování spotřebovává staminu, která se vizualizuje pomocí UI lišty. Mechanika výdrže zahrnuje postupné snižování staminy při sprintu a automatickou regeneraci po čtyřech sekundách od ukončení běhu.

Skok je implementován pomocí AddForce funkce s ForceMode.Impulse, což aplikuje okamžitou sílu směrem nahoru. Po skoku se nastaví flag readyToJump na false a po půl sekundě se znovu aktivuje, což vytváří cooldown mezi skoky.

Kontrola rychlosti je implementována pomocí výpočtu horizontální rychlosti (bez Y složky) a omezení na maximální hodnotu moveSpeed. Pokud rychlost překročí limit, vynásobí se normalizovaný vektor rychlosti maximální hodnotou.

3.2 Systém střelby s raycast technologií

Systém střelby využívá raycast, což je metoda pro vystřelení neviditelného paprsku do scény a detekci kolizí. Při stisknutí levého tlačítka myši se z pozice kamery vystřelí paprsek směrem, kam hráč míří.



Obrázek 3: Hlavní menu hry s třemi tlačítky

Raycast se provádí pomocí Physics.Raycast funkce, která vrací true, pokud paprsek zasáhne nějaký objekt. Funkce také vrací RaycastHit strukturu obsahující informace o kolizi včetně pozice dopadu, normály povrchu a reference na zasažený objekt.

Pokud raycast zasáhne objekt s tagem "Enemy", zavolá se na tomto objektu metoda TakeDamage s parametrem 1, což představuje škodu. Vizuální efekt výstřelu je implementován pomocí LineRenderer komponenty, která vykreslí linii od hlavně zbraně k bodu dopadu nebo maximálně 100 jednotek daleko.

Zvukový efekt je implementován tak, že se při každém výstřelu vytvoří dočasný GameObject s AudioSource komponentou. Tento objekt se automaticky zničí po pře-

hrání zvuku pomocí Destroy funkce s časovým parametrem.

Cooldown mezi výstřely je implementován pomocí proměnné readyToShoot, která se nastaví na false po výstřelu a po 0.6 sekundách se pomocí Invokace funkce znovu aktivuje.

3.3 Umělá inteligence zombie s NavMesh

Umělá inteligence zombie je postavena na Unity NavMesh systému. Každý zombie má připojen NavMesh Agent komponentu, která se stará o navigaci k cíli a automatické vyhýbání se překážkám.

```
// Find a random position on the NavMesh within respawnDistance of the player
private Vector3 RandomNavMeshPosition()
{
    Vector3 randomDirection = Random.insideUnitSphere * respawnDistance;
    randomDirection += playerTransform.position;

    NavMeshHit hit;
    if (NavMesh.SamplePosition(randomDirection, out hit, respawnDistance, NavMesh.AllAreas))
    {
        RaycastHit terrainHit;

        // Raycast downwards to find the ground position
        if (Physics.Raycast(hit.position + Vector3.up * 100f, Vector3.down, out terrainHit, 200f, LayerMask.GetMask("whatIsGround")))
        {
            return terrainHit.point;
        }
        else
        {
            Debug.LogWarning("Sampled position not on ground: " + hit.position);
        }
    }
    else
    {

```

Obrázek 4: Kód AI navigačního systému zombie

V Update metodě se každý snímek vypočítá vzdálenost mezi zombie a hráčem pomocí Vector3.Distance funkce. Pokud je vzdálenost větší než 1.7 jednotky, zombie se přepne do režimu pronásledování. V tomto režimu se nastaví cíl NavMesh agenta na pozici hráče pomocí agent.SetDestination a aktivuje se animace chůze.

Pokud se zombie dostane blíže než 1.7 jednotky, přepne se do útočného režimu. Deaktivuje se animace chůze a aktivuje se animace útoku. V útočném režimu zombie každé dvě sekundy způsobí hráči škodu 10 životů.

Systém zpomalování je implementován pomocí proměnné decelerationFactor, která se počítá jako funkce vzdálenosti od minimální zastavovací vzdálenosti. Tato hodnota se vynásobí s rychlostí agenta, což vytváří plynulé zpomalení při přibližování k cíli.

Při smrti zombie (když životy klesnou na nulu nebo níž) se přehraje animace smrti, deaktivuje se collider a rychlost se nastaví na nulu. Po třech sekundách se objekt zombie zničí a na náhodné pozici na mapě se vytvoří nový.

Náhodná pozice pro respawn se hledá pomocí NavMesh.SamplePosition funkce, která najde nejbližší bod na NavMesh v určité vzdálenosti od hráče. Toto zajišťuje, že se zombie vždy vytvoří na choditelném terénu.

```

// Method to apply damage to the zombie
public void TakeDamage(float amount)
{
    health -= amount;

    if (health <= 0f && !dead)
    {
        dead = true;
        animator = GetComponent<Animator>();
        zombieCollider = GetComponent<Collider>();
        zombieCollider.enabled = false;
        animator.SetTrigger("Death");
        navMeshAgent.speed = 0;

        Destroy(this.gameObject, 3f);
        if (Random.Range(0f, 1f) <= 0.2f)
        {
            LifeDisplay.currentHp = Mathf.Min(LifeDisplay.currentHp + 5, LifeDisplay.maxHp);
        }

        scorehighscore.Kill();
    }
}

```

Obrázek 5: Kód metody TakeDamage zombie

Po každých pěti zabitých zombie se rychlost všech nepřátel zvýší o 0.2 jednotky, což implementuje progresivně rostoucí obtížnost. Tato mechanika je implementována v metodě Kill, která se volá při smrti zombie.

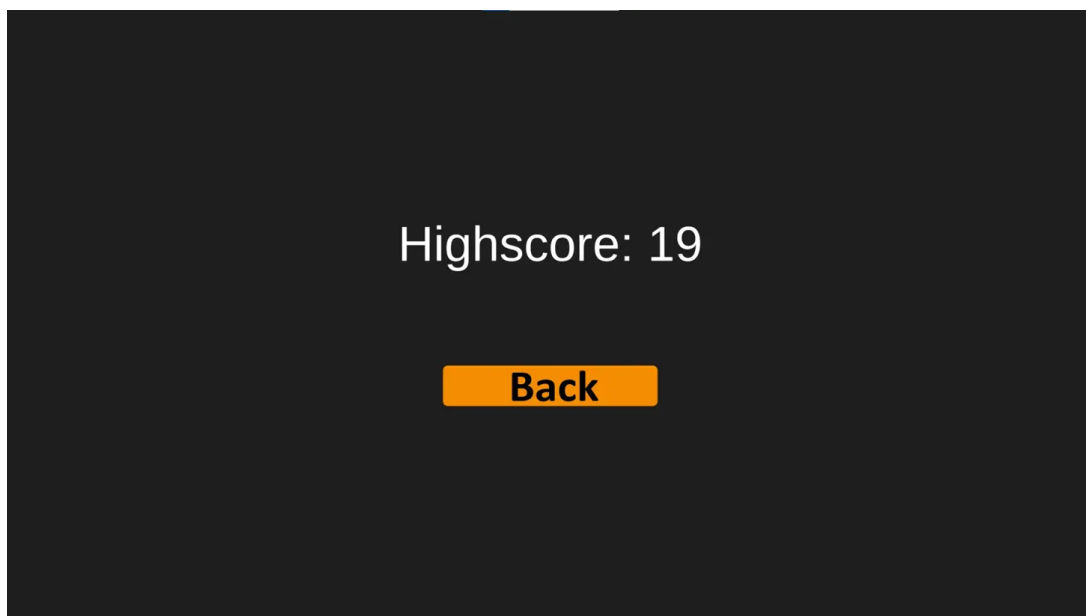
3.4 Systém skóre a scén

Hra obsahuje čtyři základní scény implementované pomocí Unity Scene Management systému. Hlavní menu obsahuje tři tlačítka vytvořená v Unity UI systému. Každé tlačítko má připojený OnClick event, který volá příslušnou metodu při kliknutí.

Skóre systém je implementován pomocí statické třídy, která uchovává aktuální skóre během hry. Za každého zabitého zombie získává hráč jeden bod, který se přičte k celkovému skóre. Nejvyšší dosažené skóre je uloženo pomocí PlayerPrefs.SetInt funkce, což je Unity systém pro persistentní uložení dat.

Při načítání scény se nejprve nastaví časová prodleva 2 sekundy pomocí yield return new WaitForSeconds, během které se provede kompletní načtení světa. Následně se deaktivuje kurzor myši pomocí Cursor.lockState a Cursor.visible, což zajistí, že kurzor bude uzamčen doprostřed obrazovky.

Fade in efekt je implementován pomocí UI Image komponenty s černou barvou, jejíž alfa hodnota se postupně snižuje od 1 do 0. Tento efekt dává čas na dokončení všech inicializačních úprav a vytváří plynulý přechod do hry.



Obrázek 6: Scéna zobrazující nejvyšší dosažené skóre

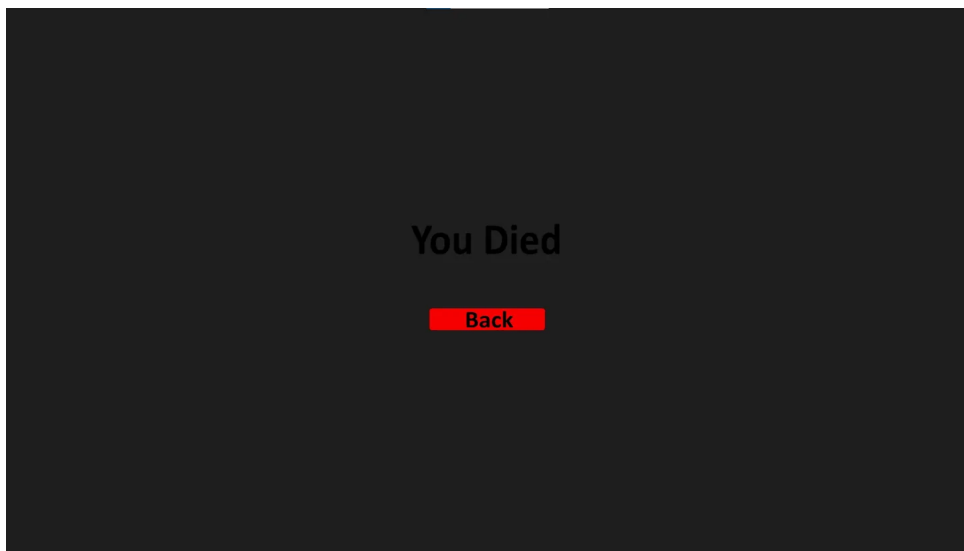
3.5 Tvorba herního světa

Prvně jsem mapu herního světa vytvořil v Blenderu, ale v průběhu práce na projektu jsem přešel na Unity Terrain systém. Nejprve jsem vytvořil terén o rozměrech 1000x1000 jednotek a pomocí Raise/Lower Terrain nástroje jsem modeloval kopce a vyvýšeniny. Mapa byla moc velká a omezil jsem na 500x500 jednotek. Dál jsem nastavil bordery na krajích mapy, aby ani hráč ani nepřítel se nespawnuli nebo nevypadli z mapy.

Textury byly aplikovány pomocí Paint Texture nástroje. Použil jsem dvě vrstvy - základní travnatou texturu a druhou vrstvu s cestičkami a hlínou. Stromů jsem rozmístil přibližně pět tisíc pomocí Mass Place Trees funkce, která umožňuje nastavit hustotu, velikost a rotaci stromů.

2D tráva s billboard efektem byla přidána pomocí Paint Details nástroje. Billboard efekt znamená, že tráva se vždy otáčí směrem ke kameře, což je méně náročné na výkon než plné 3D modely. Kameny, houby a keře jsem rozmístil ručně pomocí Place GameObject nástroje.

Mlha byla implementována pomocí Unity Lighting systému. Nastavil jsem hustotu mlhy na 0.02, barvu na světle modrou a režim na Exponential Squared. Mlha vytváří atmosféru a zároveň omezuje viditelnost, což snižuje počet renderovaných objektů. Díky tomu jde hra více plynule.



Obrázek 7: "You Died" scéna zobrazená po smrti hráče

3.6 Uživatelské rozhraní

UI bylo vytvořeno pomocí Unity UI systému (Canvas). Všechny UI prvky jsou potomky Canvas objektu, který je nastaven na Screen Space - Overlay mód.



Obrázek 8: In-game UI zobrazující životy, staminu a skóre

Stamina bar je implementován pomocí UI Image komponenty s typem Filled. Fill Amount se mění od 0 do 1 podle aktuální hodnoty staminy. Barva je nastavena na modrou a pozice v levém dolním rohu.

Životy jsou zobrazeny jako Text komponenta umístěná nad ikonou srdce. Ikona srdce je UI Image komponenta s importovanou texturou. Skóre je implementováno jako Text komponenta v horní části obrazovky. Crosshair je malý bílý UI Image umístěný přesně uprostřed obrazovky s rozměry 10x10 pixelů.

4 Výsledky řešení a výstupy

4.1 Splnění cílů práce

Všechny předem definované cíle práce byly úspěšně splněny. Podařilo se vytvořit plně funkční 3D survival hru s následujícími implementovanými funkcemi:

Pohybový systém zahrnuje plynulý pohyb pomocí WASD kláves, sprintování se systémem výdrže, skok s cooldownem, krčení pro zpomalení pohybu, a realistic-kou fyziku pomocí Rigidbody. Systém správně detekuje, zda je hráč na zemi nebo ve vzduchu, a podle toho upravuje aplikované síly.

Systém střelby využívá raycast technologii pro přesnou detekci zásahů. Každý výstřel má vizuální reprezentaci pomocí LineRenderer a je doprovázen zvukovým efektem. Cooldown mezi výstřely zajišťuje vyváženou hratelnost.

Umělá inteligence zombie je implementována pomocí NavMesh systému a poskytuje inteligentní navigaci po mapě s automatickým vyhýbáním se překážkám. Zombie mají dva režimy chování - pronásledování a útok. Systém progresivní obtížnosti zvyšuje rychlost zombie po každém druhém zabití nepřátel.

Uživatelské rozhraní zobrazuje všechny klíčové informace včetně životů, výdrže, skóre a zaměřovače. Design je minimalistický a nepřekáží hráči při hraní.

Systém scén obsahuje čtyři funkční scény - hlavní menu, herní scénu, highscore scénu a death scénu. Navigace mezi scénami je intuitivní a plynulá.

4.2 Vizuální a audio stránka

Grafická stránka hry využívá low-poly stylu, který je výkonově nenáročný a zároveň esteticky příjemný. Herní prostředí obsahuje více než deset tisíc stromů, stovky keřů a kamenů, a dynamickou mlhu pro atmosféru.

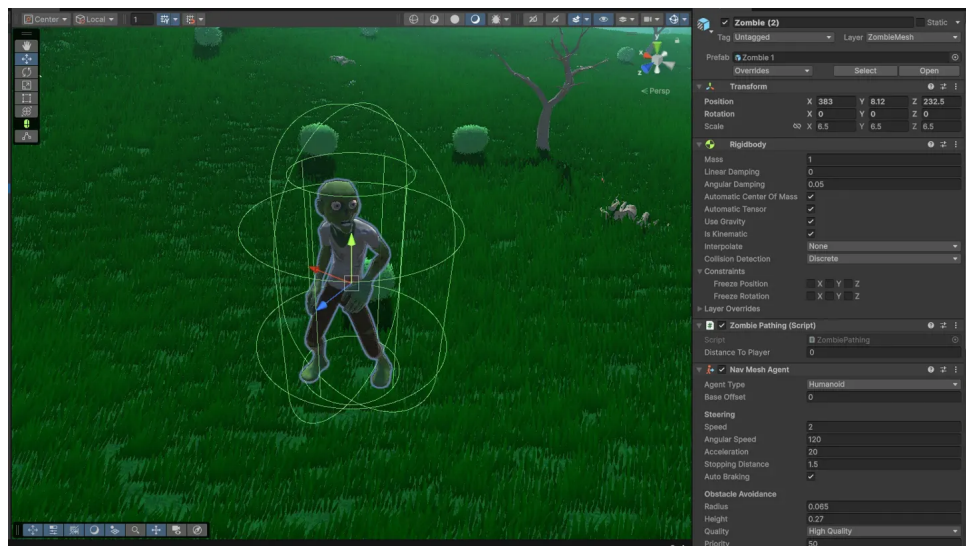
Zombie model je detailní s texturovanou kůží a oblečením. Model obsahuje tři animace, které plynule přecházejí mezi sebou pomocí Animator Controlleru. Laserová pistole zapadá do celkového vizuálního stylu hry.

Zvuková stránka zahrnuje efekt laserového výstřelu, který odpovídá futuristickému charakteru zbraně. Zvuk se přehrává při každém výstřelu bez nežádoucího překrývání díky dynamickému vytváření audio objektů.

4.3 Výkon a optimalizace

Hra je optimalizována pro plynulý běh na běžných počítačích. Použití low-poly modelů výrazně snižuje počet polygonů, které je potřeba renderovat. Mlha omezuje viditelnost a tím pádem počet renderovaných objektů v dálce.

Billboard efekt u trávy znamená, že místo tisíců 3D modelů se renderují pouze



Obrázek 9: Zombie model v různých prostředích

2D sprity, které se otáčejí směrem ke kameře. NavMesh systém je optimalizován Unity a poskytuje výkonnou navigaci bez nutnosti výpočtu pathfindingu v každém snímku.

Dynamické vytváření a ničení objektů (jako audio efekty nebo zombie při respa-wnu) zajišťuje, že ve scéně nejsou zbytečné objekty, které by zatěžovaly systém.

4.4 Testování a ladění

Během vývoje probíhalo průběžné testování všech mechanik. Unity Console poskytoval okamžitou zpětnou vazbu při chybách. Debug.Log zprávy byly použity pro sledování hodnot proměnných a stavu hry.

Testování zahrnovalo kontrolu kolizí, správného chování AI při různých situacích, fungování UI elementů a persistentního ukládání skóre. Všechny identifikované chyby byly postupně opraveny.

4.5 Uživatelský manuál

Ovládání hry:

- WASD nebo šipky - pohyb postavy
- Myš - otáčení kamery a míření
- Levé tlačítko myši - střelba
- Levý Shift - sprint (spotřebovává výdrž)
- Mezerník - skok
- Levé Ctrl - krčení
- Escape - návrat do menu

Cíl hry: Přežít co nejdéle a dosáhnout co nejvyššího skóre eliminací zombie. Za každého zabitého zombie získáte 1 bod. Každé druhé zabití zombie se rychlost nepřátel zvyšuje.

Herní mechaniky: Zombie vás budou pronásledovat a při kontaktu způsobovat škodu. Každý zombie má 3 životy a je potřeba ho zasáhnout třikrát. Hráč začíná se 100 životy. Po smrti se zobrazí scéna "You Died" a můžete se vrátit do menu.

Závěr

V rámci této závěrečné práce se mi podařilo vytvořit plně funkční 3D survival hru v herním enginu Unity. Všechny předem stanovené cíle byly úspěšně splněny - implementoval jsem pohybový systém s pokročilými mechanikami, vytvořil systém střelby využívající raycast technologii, naprogramoval umělou inteligenci nepřátel pomocí NavMesh, navrhl uživatelské rozhraní a implementoval systém skóre s persistentním ukládáním.

Projekt mě naučil programovací jazyk C# a seznámil jsem se s komplexním procesem vývoje her. Během vývoje jsem se naučil s Unity Editorem, NavMesh systémem pro AI navigaci, práci s 3D modely a animacemi, návrhem uživatelského rozhraní a optimalizací výkonu hry.

Výsledná hra nabízí zábavný herní zážitek s jasným cílem a progresivně rostoucí obtížností. Systém skóre motivuje hráče k dosahování lepších výsledků a persistentní ukládání nejvyššího skóre umožňuje porovnávání výkonů mezi jednotlivými herními sezeními.

Hra je plně hratelná a obsahuje všechny klíčové prvky survival žánru. Grafická stránka využívá low-poly stylu, který je esteticky příjemný a zároveň výkonově nenáročný. Umělá inteligence zombie poskytuje zajímavou výzvu díky inteligentní navigaci a dvěma režimům chování.

Do budoucna plánujeme rozšířit hru o globální žebříček s online databází, více map s různými prostředími, carousel zbraní s různými vlastnostmi, více úrovní obtížnosti a další vylepšení jako power-upy. Realizace tohoto projektu mi ukázala, jak složité je vytváření her, a rozšířila mé obzory v oblasti herního vývoje.

Seznam použitých informačních zdrojů

1. Unity Learn. UNITY TECHNOLOGIES. *Unity Learn* [online]. 2005 [cit. 2026-01-06]. Dostupné z: <https://learn.unity.com>
2. Unity Documentation. *Unity Manual* [online]. [cit. 2026-01-06]. Dostupné z: <https://docs.unity3d.com/Manual/>
3. EROKIA. *Plasma - Lazer Pistol Gun Shot 1* [online]. Freesound.org, 2018 [cit. 2026-01-12]. Dostupné z: <https://freesound.org/people/Erokia/sounds/427396/>
4. *Pistol* [online]. Poly Pizza, 2021 [cit. 2026-01-06]. Dostupné z: <https://poly.pizza/m/J3i9KDQ3kt>
5. *Low Poly Zombie* [online]. Sketchfab, 2019 [cit. 2026-01-06]. Dostupné z: <https://sketchfab.com/3d-models/low-poly-zombie-game-animation-9f8f0885b2f94c6890c4d>
6. *Lowpoly Environment - Nature Free - MEDIEVAL FANTASY SERIES* [online]. Unity Asset Store, 2023 [cit. 2026-01-12]. Dostupné z: <https://assetstore.unity.com/packages/3d/environments/lowpoly-environment-nature-free-medieval-fa>
7. Microsoft Learn. *C# Documentation* [online]. [cit. 2026-01-06]. Dostupné z: <https://learn.microsoft.com/cs-cz/dotnet/csharp/>

Seznam obrázků

1	Unity Editor s nastaveným zombie objektem	4
2	Kód implementující pohybový systém hráče	6
3	Hlavní menu hry s třemi tlačítky	7
4	Kód AI navigačního systému zombie	8
5	Kód metody TakeDamage zombie	9
6	Scéna zobrazující nejvyšší dosažené skóre	10
7	"You Died"scéna zobrazená po smrti hráče	11
8	In-game UI zobrazující životy, staminu a skóre	11
9	Zombie model v různých prostředích	13