GYMNÁZIUM, STŘEDNÍ PEDAGOGICKÁ ŠKOLA, OBCHODNÍ AKADEMIE A JAZYKOVÁ ŠKOLA s právem státní jazykové zkoušky ZNOJMO, příspěvková organizace

669 02 Znojmo, Pontassievská 3 • tel.: 515158101 • fax: 515225230 • e-mail: info@gpoa.cz • www.gpoa.cz, IČ:49438816

Maturitní práce

Žák: David Beneš
Třída: E4.C
Školní rok: 2017/2018
Obor vzdělání: Informační technologie

Název maturitní práce: Tvorba počítačové hry

Vedoucí maturitní práce: Mgr. Milička Tomáš
Oponent maturitní práce:

Sociální partner: GYMNÁZIUM, STŘEDNÍ PEDAGOGICKÁ ŠKOLA, OBCHODNÍ AKADEMIE A JAZYKOVÁ ŠKOLA s právem státní jazykové zkoušky ZNOJMO, příspěvková organizace

Práce odevzdána dne: (datum)

Podpis žáka: (podpis)

Prohlášení

Prohlašuji, že jsem maturitní práci vypracoval samostatně pod vedením Mgr. Tomáše Miličky a uvedl v seznamu literatury a zdrojů veškerou použitou literaturu a další informační zdroje.

Místo datum

David Beneš

Anotace

Maturitní práce se zaměřuje na programování v jazyce C# v herním enginu Unity. V práci je popsán vývoj hry a veškeré problémy, které s vývojem souvisí. Pro editaci skriptů je použit program Visual Studio Community 2017. Pro obrázky v UI byl použit grafický program Krita. Veškerá ostatní práce spočívala v programu Unity Editor, kde byly vytvořeny všechny úrovně a animace postav.

Klíčová slova

hra, csharp, unity, programování, visual studio

Anotation

Graduation project is aimed at programming in C# language in game engine Unity. This paper describes entire development of the game and every problems, which are related with it. Visual studio is used to edit scripts. Krita is used to make images for the UI. Everything else is done in Unity Editor, where every level and animation is made.

Key words

game, csharp, unity, programming, visual studio

Obsah

1 Úvod

Téma jsem si vybral, protože mě vždy zajímalo jak se hry vyvíjí. Dále jsem chtěl vědět, jestli vůbec takovou hru dokážu naprogramovat. Také jsem zastáncem názoru, že programování se dá naučit jedině praxí.

1.1 Vymezení cílů

Mým hlavním cílem je naprogramovat hru pomocí jazyka C#. Neměla by být moc obtížná na naučení, ale měla by být těžká na pokoření [1]. Game design by měl být jednoduchý a přehledný, ať se hráč jednoduše vyzná kde se právě nachází. Hra by se měla zaměřovat na schovávání se před nepřáteli, a ne na jejich zabíjení.

2 Využité programy

2.1 Unity

2.1.1 Představení

Unity je herní engine. Je používaný pro vývoj jak indie (nezávislých), tak AAA her. Je v něm možná naprogramovat hru v jazyce C#, který je rozšířen o mnoho Unity knihoven, ale i v JavaScriptu.

2.1.2 Důvod použití

Unity jsem si vybral, protože se mi hned na první pohled zalíbilo. Má krásný a jednoduchý design. Také znám spoustu her, které jsou za pomocí Unity enginu vytvořené. Velikým důvodem pro výběr byla také dostupnost v jazyce C#, což ne všechny herní enginy nabízejí.

2.1.3 Výhody

- Jednoduchost
- Je zdarma pro osobní i komerční účely (do $100k)
- Dostupnost jazyka C#

2.1.4 Nevýhody

- Pro úplné nástroje je třeba zaplatit

2.1.5 Alternativy

- Unreal Engine
  - Od společnosti Epic Games
  - Více se zaměřuje na systém akce a reakce, než se zaměřuje na programování
- CryEngine
  - Od společnosti Crytech

2.2 Visual Studio

2.2.1 Představení

Pro upravování C# skriptů jsem použil program Visual Studio Community 2017 [9]. Tento program vytvořil microsoft, proto má skvělou podporu jazyka C#.

2.2.2 Důvod použití

Ve Visual Studiu jsme pracovali i v hodinách programování ve škole, takže jsem měl k programu velice blízko. Dalším důvodem bylo to, že Unity velice dobře pracuje s tímto programem. Nabízí snippets [12], proto je mnohem jednodušší a rychlejší kód psát. Je tedy mnohem jednodušší soustředit se na to, co je důležité a ne na pamatování si všech metod a tříd, které unity poskytuje.

2.2.3 Výhody

- Je zdarma pro osobní i komerční účely pro jedince
- Dostupnost jazyka C#
- Jednoduché debugování

2.2.4 Nevýhody

- Není multiplatformní (neexistuje verze pro linux)

2.2.5 Alternativy

- monodevelop
- atom
- vs code

3 Průběh vývoje

3.1 Výběr žánru

Na začátku jsem musel vymyslet v jakém duchu budu hru vyvíjet. Jelikož mám velice rád stealth hry [13], rozhodl jsem se ji vytvořit v tomto duchu. Dále jsem musel vymyslet, jaký styl úrovní budu vytvářet. Jelikož jsem dlouho měl jen slabý počítač, mohl jsem hrát pouze nenáročné hry většinou indie vývojářů. Mnoho z nich bylo tzv. dungeoncrawler [10], proto jsem se rozhodl jít právě touto cestou. Nechtěl jsem, aby hráč mohl zabíjet neviné strážce, a tak ve hře nemá k dispozici žádné zbraně. Pro záživnější průběh jsem se rozhodl zahrnout do hry nějaké superschopnosti, které by zbraně nahradili.

3.2 Ovládání postavy

Hráč ovládá postavu pomocí základních WASD tlačítek. Pomocí myši míří a rozhlíží se po úrovni. Pro pohyb jsem využil Vector3 [3] a to tak, že jsem k pozici postavy přičetl jednotkový vektor a vynásobil rychlostí, kterou jsem si vybral. Pro otáčení hráče ke kurzoru jsem prvně musel zjistit, kde je kurzor. To jsem udělal pomocí kamery, která je na scéně. Ta dokáže vypočítat, kam na jaký bod kurzor míří. Aby se postava nedívala do země, přičetl jsem výšku postavy k tomuto bodu. Pro samotné otočení postavy k bodu jsem musel využít quaternion [4] Ten sice zjistí, jak daleko se musí postava otočit, ale poté je nutno ho převést na tzv. eulerAngles [5] Díky tomu pak víme, o kolik se na jednotlivých osách postava musí natočit.

3.3 Schopnosti

Při výběru schopností jsem se nechal inspirovat ze stealth her a moba her. Hlavně pak Dishonored a Dota2.

3.3.1 Blink

Blink je teleportace na krátkou vzdálenost. Pomáhá hráči k rychlému přesunu. Prvně jsem se rozhodl vytvořit vizuální zobrazení pro hráče. Vytvořil jsem čáru, které začala v postavě a končila na kurzoru hráče (pro získání kurzoru jsem využil stejný postup jako při otáčení postavy). Na konci čáry jsem přidal kostku, aby hráč věděl, kam se přesune. Poté jsem řešil maximální vzdálenost teleportu. Vytvořil jsem si proměnou, která uchovávala desetinné číslo, které určuje maximální vzdálenost. Pomocí IF podmínky jsem zjistil, jestli vzdálenost postavy od konce čáry je delší než je tato proměnná. Pokud ano zkrátím čáru na maximální vzdálenost ve směru kurzoru. Poté jsem musel zabránit teleportaci skrz zdi. Prvně jsem zjistil, jestli je na konci čáry zeď pomocí Physics.Raycast [6] Pokud ano, tak čáru zkrátil na vzdálenost hráče od této zdi. Nakonec jsem musel vyřešit samotný teleport. To bylo velice jednoduché, protože stačilo nastavit pozici postavy na pozici, která je na konci čáry.

3.3.2 Neviditelnost

Pro neviditelnost jsem použil globální proměnnou, kterou jsem si vytvořil. Tu jsem pak předával nepříteli, aby "nevnímal" hráče. Těžší bylo zajistit, aby neviditelnost trvala nějakou dobu a pak se hráč opět zviditelni. Vytvořil jsem while cyklus a proměnou, která uchovávala čas, který může být hráč neviditelný. Každým průchodem cyklu jsem od této proměnné odečetl sekundu pomocí Time.Deltatime [7] Dokud byla proměnná větší jak 0, tak byl hráč neviditelný. Aby hráč věděl, kdy je neviditelný musel jsem změnit barvu postavy. Prvně jsem získal materiál postavy a pak nastavil průhlednost nastavil na 1/3. Až se hráč opět zviditelnil, nastavil jsem zpět hodnotu 1.

3.3.3 Cooldown

Aby hráč nemohl být neviditelný donekonečna, a nemohl se stále teleportovat musel jsem vytvořit cooldown. Pro cooldown jsem využil podobný cyklus jako pro neviditelnost. Jediné co jsem musel přidat byla proměnná, která uchovává, jestli hráč muže schopnost použít. Před cyklem jsem ji nastavil na false a po cyklu opět na true. Aby šlo vidět, kolik času zbývá, rozhodl jsem se ho zakomponovat do UI.

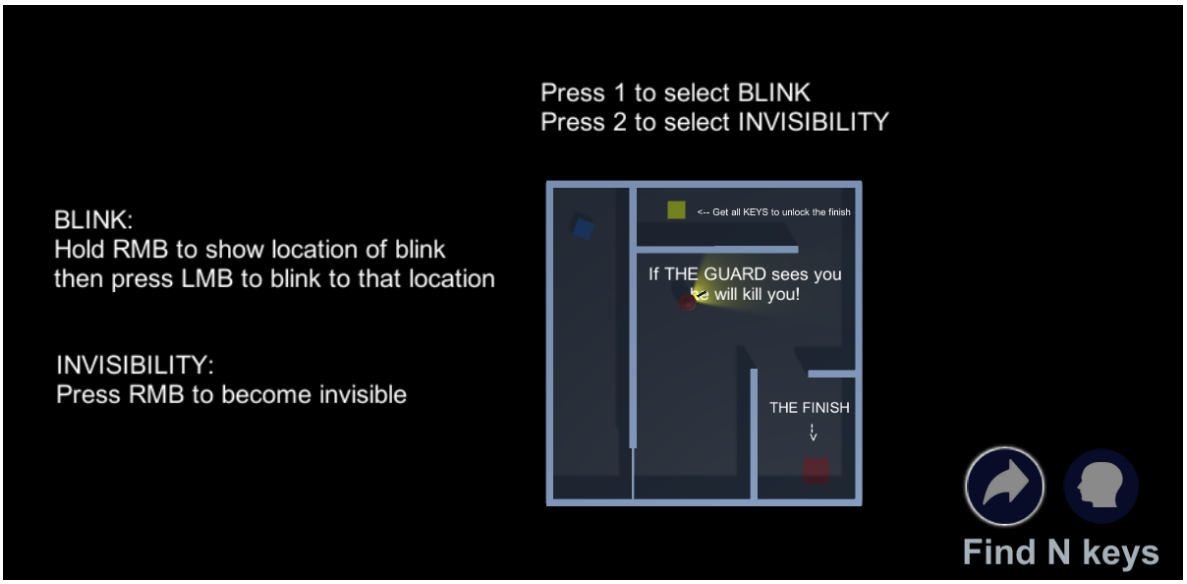
3.4 Nepřítel

Hned na začátku jsem věděl, že nepřítel nebude moct umřít. Tím se mi uvolnila práce na životech nepřítele. Aby hra neměla v sobě neměla RNG faktor (REFERENCE), nepřítel stále následuje dokolečka jednu cestu. Trasa je seskládána z několika bodů, na které musí nepřítel dojít. Jakmile k nim dojde, otočí se k dalšímu atd... Ve skriptu jsem musel v poli uchovávat místo těchto bodů, opět jsem k tomu využil vektorů. Když jsem měl vektory uložené, mohl jsem využít funkcí unity a přesunout nepřítele k bodu. Aby to ovšem nevypadalo špatně, bylo nutné zařídit otáčení nepřítele k danému bodu. Naštěstí v knihovnách Unity jsou i goniometrické funkce, díky tomu jsem pomocí dvou vzdáleností, které bod svírají, mohl zjistit o kolik stupňů se nepřítel musí otočit. Když už byl nepřítel schopný chodit po trase, chtěl jsem z něj pro hráče udělat nějakou hrozbu. Rozhold jsem se mu přidat zbraň. Kvůli tomu jsem musel naznačit, kam nepřítel vidí. To jsem udělal pomocí světla, které jsem nasadil na nepřítele. V kódu jsem pak nastavil, jak daleko a jaký úhel může nepřítel vidět. Abych dal hráči najevo, že ho nepřítle spatřil, světlo mění barvy. Prvně jsem ale potřeboval zjistit, jestli hráč vůbec žije. Pokud žije a pokud vstopí do zorného pole nepřítele, světlo zčervená a nepřítel hráče zastřelí. Samozřejmě bylo nutné přidat nějakou časovou odchylku, aby hráč měl větší šanci utéct. Nakonec nepřítel vystřelí, hráč umírá a musí restartovat úroveň.

3.5 Úrovně

3.5.1 Cíl úrovní

Hlavní hráčův úkol je posbírat všechny klíče, které jsou po úrovni rozmístění. Při tom se musí schovávat před zrakem nepřátel. Sebrání všech klíčů však není konec, hráč se stále musí dostat do cíle, který právě otevřel. Jakmile hráče nepřítel spatří, zastřelí ho a tím se úroveň restartuje.



Obrázek 3.1: Úvodní úroveň

3.5.2 Přechod do další úrovně

Aby přechod do další úrovně vypadal pěkně, našel jsem si na Youtube návod na animace. Aby to udělalo fade effect, musel jsem vytvořit 2D plochu. Když hráč došel na konec úrovně, pomalu se přechází na tuto plochu a až po té se přepne úroveň. V C# je přechod zajištěn změnou průhlednosti plochy v cyklu. Pomocí proměné DeltaTime se pak dá každý průchod cyklem spozdit o určitý čas.

3.5.3 Otevírání a zavírání dveří

Pro hezčí vzhled úrovní jsem přidal dveře. Když se k nim hráč přiblíží (vstoupí do jejich collideru) dveře se otevřou. Když se od nich vzdáli zase se zavřou.

3.6 Menu

Všechna grafika, která se v menu nachází byla vytvořena díky samotnému Unity Enginu. Podobně jako jsem udělal UI, jsem také udělal menu. Jediné co se změnilo je, že jsem přidal tlačítka, které mají nějakou funkci.

3.6.1 Základní menu

Abych mohl vytvořit menu, musel jsem zjistit, jak v Unity funguje 2D grafika. Pro jednotlivá tlačítka v menu jsem musel vytvořit jednotlivé funkce v C# skriptu. Pro přechod do dalšího menu se jednoduše načte jiná Unity scéna. Každá scéna má své jméno a podle toho se pak dá v C# zavolat.kk

Obrázek 3.2: Pause menu

3.6.2 Zapauzování hry

Zapauzování hry bylo jednodušší, než jsem ze začátku čekal. Unity totiž udržuje v promněné GameObject.activeSelf, jestli je daný objekt aktivní nebo ne. To jsem využil tak, že jsem si načetl objekt, na kterém je ckript nasazen a pojmenoval jsem si ho UI. Pomocí `UI.activeSelf` zjistil, jestli je menu aktivní nebo ne. Potom jsem při zmáčknutí Escape jen znegoval hodnotu activeSefl. Tím jsem měl zajištěno ukázání a schování pauzovacího menu. Poté jsem ještě potřeboval samotnou hru zmrazit. To jsem zařídil pomocí `Time.timeScale`. Tato proměnná udržuje, jak rychle běží čas ve hře. Základní hodnota je 1, pro zapauzování jsem ji změnil na 0 a naopak.

```
// NORMAL MODE
else
{
    //blink range
    if (Vector3.Distance(blinkCube.transform.position, transform.position) > blinkRange)
    {
        lr.SetPosition(1, (blinkCube.transform.position - transform.position).normalized * blinkRange + transform.position);
        blinkCube.transform.position = lr.GetPosition(1);
    }

    //collision
    RaycastHit hit;
    float maxDistOfColDetection = Vector3.Distance(transform.position, blinkCube.transform.position);
    if (Physics.Raycast(transform.position, transform.TransformDirection(Vector3.forward), out hit, maxDistOfColDetection, blinkObstacleMask))
    {
        lr.SetPosition(1, hit.point);
        blinkCube.transform.position = hit.point;
    }

    if (Input.GetMouseButtonDown(0) && canBlink)
    {
        //blink to position
        transform.position = blinkCube.transform.position;

        //set player to visible
        GetComponent<Renderer>().material.color = new Color(playerColor.r, playerColor.g, playerColor.b, 1f);
                        Game.instance.durationInvisibilityImage.fillAmount = 0f;
                        Player.isInvisible = false;

        //play audio and animation
        Audio.instance.PlaySound(blinkSound, transform.position);
        anim.Play("BlinkAnimation");

        //change of color on cooldown
        blinkCube.GetComponent<Renderer>().sharedMaterial.color = colorOfBlinkCubeOnCooldown;
        canBlink = false;

                        float c = cooldown;
                        while (c > 0f)
                        {
                                c -= Time.deltaTime;
                                Game.instance.cooldownBlinkText.text = (c > 0) ? Mathf.CeilToInt(c).ToString() : null;
                                Game.instance.cooldownBlinkImage.fillAmount = c / cooldown;
                                yield return null;
                        }
        canBlink = true;
        blinkCube.GetComponent<Renderer>().sharedMaterial.color = colorOfBlinkCube;
    }
}
```

Obrázek 3.3: Teleportace postavy

```
//NORMAL MODE
else if (Input.GetMouseButtonDown(1) && !invisible)
{
    //change color to "invisible"
    invisible = true;
    Player.isInvisible = true;
    GetComponent<Renderer>().material.color = new Color(playerColor.r, playerColor.g, playerColor.b, 1 / 3f);
    GetComponent<Renderer>().shadowCastingMode = UnityEngine.Rendering.ShadowCastingMode.Off;

                //cooldown
                float c = cooldown;
                while (c > 0f)
                {
                        c -= Time.deltaTime;
                        //duration of invisibility
                        float d = duration;
                        while (d > 0f && Player.isInvisible)
                        {
                                d -= Time.deltaTime;
                                Game.instance.durationInvisibilityImage.fillAmount = d / duration;
                                yield return null;
                        }
                        //making player visible
                        Player.isInvisible = false;
                        GetComponent<Renderer>().material.color = new Color(playerColor.r, playerColor.g, playerColor.b, 1f);
                        GetComponent<Renderer>().shadowCastingMode = UnityEngine.Rendering.ShadowCastingMode.On;

                        Game.instance.cooldownInvisibilityText.text = (c > 0) ? Mathf.CeilToInt(c).ToString() : null;
                        Game.instance.cooldownInvisibilityImage.fillAmount = (c > 0f) ? c / cooldown : 0f;
                        yield return null;
                }

                invisible = false;
}
```

Obrázek 3.4: Neviditelnost postavy

```
//rigidbody uses fixedupdate
private void FixedUpdate()
{
        myRigidbody.MovePosition(myRigidbody.position + velocity * Time.fixedDeltaTime);
}
```

Obrázek 3.5: Přesun postavy

4 Problémy vývoje

4.1 Úvod

Jelikož jsem začátečník v programování a zároveň v enginu Unity, narazil jsem na spoustu problémů. Pár větších bych teď chtěl popsat.

4.2 Pohyb a rotace postavy

Při pohybu jsem musel upravovat rychlost. Po té jsem narazil na problém při kolizích postavy a země či zdí. Postava stále propadávala. Zjistil jsem, že to bylo ve velikosti collideru země, o něco málo jsem o zvětšil a vše bylo v pořádku. Při rotaci to ovšem bylo horší. Chtěl jsem, aby se postava otáčela ke kurzoru. Byl zde problém zjištění místa kurzoru. Ten jsem docela rychle vyřešil pomocí některých metod, které unity poskytuje. Po té se ale postava stále koukala do země, takže jsem se nemohl normálně pohybovat. Pomocí youtube videí jsem zjistil, že se dá nalézt kurzor elegantněji. To mi sice pomohlo, ale stále se postava dívala trošku jinam, než měla. Nakonec jsem to vyřešil přičtením této odchylky a vše funguje normálně.

4.3 Nepřátelé

Stejně jako u hráče bylo nutné nastavovat rychlost pohybu, stejně nutné ovšem byla i rychlost otáčení. S otáčením jsem narazil na několik problémů. Prvně se nepřítel otáčel nejdelší cestou. To se dalo vyřešit pomocí podmínky, která určovala, aby se otáčel o úhel ostrý. Dalším problémem bylo, že se nepřítel nedokázal 'dotočit'. Ve skutečnosti zbýval tak malý úhel, že se s tím unity nevypořádalo, a tak sebou nepřítel na místě škubal. Vyřešil jsem to podmínkou, která říkala, že se nepřítel přestane točit, když daný úhel bude velice malý.

4.4 Dveře

Na začátkou jsem neměl tušení, jak zařídit průchod dveřmi. Na internetu jsem našel spoustu návodů, ale nic nebylo přesně jak bych si představoval. Otevírání jsem vyřešil pomocí animace. To bohužel nestačilo, protože jsem musel animaci nějak spustit. Zde nastal velký problém, protože jsem nedokázal zjistit, kdy se hráč ke dveřím přiblíží. Narazit do nich nemohl, protože by to nevypadalo dobře. Nakonec mě napadlo, že bych mohl zvětšit collider dveří, to sice fungovalo, ale animace byla příliž krátká. Díky tomu, že Unity dovoluje jednoduše animace vytvářet či upravovat, hned jsem to opravil.

5 Závěr

S hrou jsem nadmíru spokojen. Získal jsem mnoho programovacích zkušeností, které by se mi jinak nenaskytly. Dokázal jsem využít již naučené techniky a k nim se naučit nové.

5.1 Zhodnocení cílů

Mnoho svých cílů jsem splnil přesně dle očekávání. Některé jsem musel upravit, aby pro hráče dávali větší smysl a hra byla hratelná. Zkušební verzi jsem poskytoval kamarádům, kteří mi poté poskytli odezvu. Díky tomu jsem například přidal FOG OF WAR [11] aby hra nebyla tak jednoduchá.

5.2 Feedback???

Hru možná budu sdílet na redditu

Seznam obrázků

Odkazy

[1] Nolan Bushnell. *Theorem: Easy to Learn, Difficult to Master*. 2018. URL: http://www.wolfsheadonline.com/bushnells-theorem-easy-to-learn-difficult-to-master.

[2] Unity Technologies SF. *Unity User Manual*. 2018. URL: https://docs.unity3d.com/Manual.

[3] Unity Technologies SF. *Unity User Manual*. 2018. URL: https://docs.unity3d.com/ScriptReference/Vector3.html.

[4] Unity Technologies SF. *Unity User Manual*. 2018. URL: https://docs.unity3d.com/ScriptReference/Quaternion.html.

[5] Unity Technologies SF. *Unity User Manual*. 2018. URL: https://docs.unity3d.com/ScriptReference/Quaternion.Euler.html.

[6] Unity Technologies SF. *Unity User Manual*. 2018. URL: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html.

[7] Unity Technologies SF. *Unity User Manual*. 2018. URL: https://docs.unity3d.com/ScriptReference/Time-deltaTime.html.

[8] Asbjørn Thirslund. *Tower Defense Tutorial*. 2018. URL: https://www.youtube.com/playlist?list=PLPV2KyIb3jR4u5jX8za5iU1cqnQPmbzG0.

[9] *Visual Studio Community 2017*. 2018. URL: https://www.visualstudio.com/vs/community.

[10] Wikipedia. *Dungeon crawl*. 2018. URL: https://en.wikipedia.org/wiki/Dungeon_crawl.

[11] Wikipedia. *Fog of war*. 2018. URL: https://en.wikipedia.org/wiki/Fog_of_war.

[12] Wikipedia. *Snippet (programming)*. 2018. URL: https://en.wikipedia.org/wiki/Snippet_(programming).

[13] Wikipedia. *Stealth game*. 2018. URL: https://en.wikipedia.org/wiki/Stealth_game.