ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Semestrální práce KIV/NET

Unity - Wolfpack Project

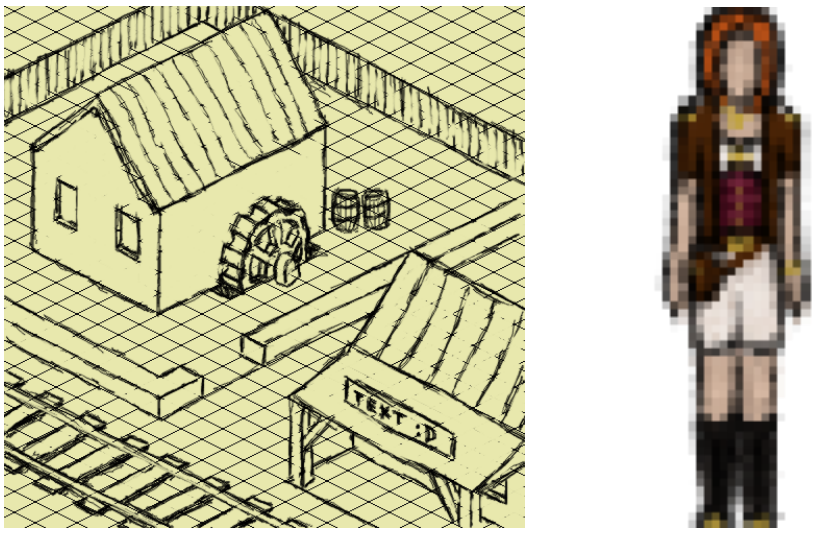
Jan Vampol
vampolj@students.zcu.cz
15. 5. 2016

1 Zadání

1.1 Popis a cíl projektu

Cílem této práce je vytvoření izometrické hry ve 2D, ve které bude hráč procházet steampunkově stylizovaným světem a jeho příběhem. Hlavní náplní hry budou souboje ve všech osmi izometrických směrech. Protivníky bude ovládat počítač.

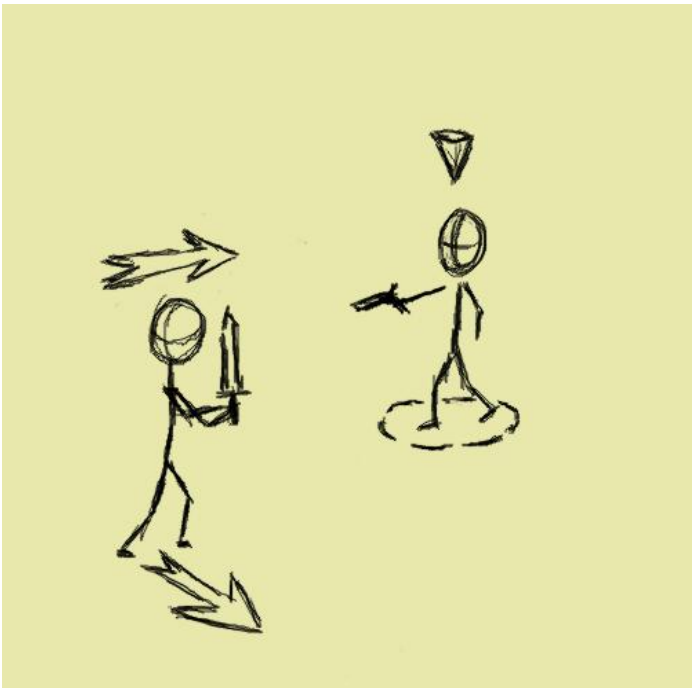
Dalším důležitým prvkem je stylizace do retro pixelovaného vzhledu s využitím omezené palety barev. To nám otevírá možnost nezaměřovat se na grafickou stránku světa, ale hlavně na jeho estetiku a stylizaci. Důraz je kladen na výběr správné barevné palety a realisticky vypadajících postavách a prostředích (v rámci steampunkového světa).



1 Izometrický pohled a příklad postavy (a barevné palety)

1.2 Náplň hry

Nejdůležitější částí hry je soubojový systém. V něm je kladen důraz na využití všech izometrických směrů. Hráč bude mít v boji možnost útoků a používání předmětů, které během hry získá. Mezi tyto patří léčivé předměty a zbraně na dálku (např. házecí nůž). Jako obraný mechanizmus bude hráči sloužit úskok ve kterémkoliv směru.



2 Soubojový systém

Další herní mechanikou bude možnost zaměřování nepřátel. Po zaměření nepřítele ho bude postava sledovat celým tělem a všechny útoky povede na něj. Tato mechanika bude využívána jak pro souboje nablízko, tak i pro boj nadálku. Zaměřování nebude nezbytné pro souboj.

Po zasažení postavy nepřátelským útokem je možné, že postava útok nevydrží a bude na krátkou chvíli omráčena (stagger). To jestli postava útoku odolá nebo ne určuje její vlastnost nazvaná Poise. Ta jednoduše určuje kolik zranění může postava dostat před tím než ránu nevydrží a dostane omráčení.

2 Analýza

K vývoji využíváme Unity2D verze 5.3.3f1. Všechny skripty, které jsou přímo spojené s problematikou hry vytváříme vlastní. Pro vytváření spritesheetů a ostatních grafických zdrojů využíváme GIMP a Spriter.

Pro ulehčení vytváření izometrických map používáme program Tiled a k němu pomocný nástroj Tiled2Unity. Pomáhá nám při vytváření základu mapy. Zbytek věcí do scén přidáváme ručně pomocí Unity Editoru.

2.1 Unity

Unity 3D je multiplatformní herní engine vyvinutý společností Unity Technologies. Byl použit pro vývoj her pro PC, konzole, mobily a web. První verze byla představena na celosvětové konferenci Applu v roce 2005. Od té doby byl rozvinut na více než patnáct dalších platforem.

Unity poskytuje možnosti vývoje pro 2D i 3D hry libovolného žánru a zaměření. Kromě grafického prostředí pro tvorbu, podporuje také tvorbu skriptů především v jazyce C# a Javascript.

2.2 Mono C#

Unity pro překlad C# skriptů používá překladač Mono. Ten je nyní už plně podporován společností Microsoft jako OpenSource implementace .NET Frameworku. V tuhle chvíli je nejvyšší verze Mono 3.8. Ta obsahuje všechny funkce C# až do verze 5.0 a již některé funkce verze 6.0.

2.3 UnityEngine

Takto je označená nejdůležitější knihovna pro programování skriptů do Unity. Zajišťuje správu scény a přístup ke všem jejím objektům. Všechny objekty ve scéně jsou uloženy ve stromové struktuře a každý uzel je schopný držet, spravovat a spouštět komponenty, které jsou k němu přidělené. Nejčastěji se toto využívá pro přidávání skriptů do objektů. Všechny funkce objektů zajišťuje třída GameObject. Všechny uzly jsou její instancí a také v sobě má několik statických funkcí pro práci s objekty.

Tato knihovna také zajišťuje správu jádra, které provádí všechny grafické a fyzikální výpočty.

Druhou nejdůležitější třídou v knihovně UnityEngine je třída MonoBehaviour. Ta je mostem mezi komponentem a spustitelným skriptem. Každý skript, který bude přidaný k objektu musí tuto třídu dědit, jinak by nebylo možné zajistit spuštění skriptu.

Všechny skripty dědící MonoBehaviour jsou během svého života mnohokrát volány jádrem Unity. Každý komponent, který je zároveň i MonoBehaviour je od chvíle vytvoření rodičovského objektu až do chvíle jeho zániku opakovaně volán pomocí mnoha metod. Tyto metody nejsou deklarované v MonoBehaviour, takže není třeba je překrývat. Za to je možné měnit hlavičku funkcí jak je libo za předpokladu, že je název funkce stejný.

2.3.1 MonoBehaviour

Unity definuje velké množství (napočítal sem 27) funkcí, které se periodicky volají za běhu aplikace. Všechny je možné prozkoumat zde. Popíšu pouze důležité a často používané.

- Awake - spustí se jednou pro každý skript jako první volaná funkce
- Start - spustí se jednou pro každý skript jako poslední funkce inicializace
- Update - volá se periodicky každý snímek
- FixedUpdate - volá se periodiky po předem daných intervalech
- OnEnable - spustí se ve chvíli, kdy se objekt stane dostupným

Důležitou součástí MonoBehaviour je také správa a volání Coroutines. Coroutine je v Unity funkce, která je schopná zastavit svojí činnost a vrátit se s tím, že činnost po určité době znovu obnoví a bude pokračovat v provádění. Taková funkce se vyznačuje tím, že její návratový typ je IEnumerator a uvnitř funkce musí být volání pozastavení s klíčovým slovem yield. Takové funkce se používají pro periodické, nebo opožděné provádění funkcí nezávisle na funkci Update.

2.3.2 GameObject

Třída GameObject je asi nejpoužívanější třídou v aplikaci. Její instance zastupují každou entitu na scéně a také dovolují správu komponentů, které jsou pod nimi uložené. Přístup k těmto komponentům se provádí hlavně pomocí funkcí GetComponent<T>() a GetComponents<T>(). Liší se pouze v tom, že první z nich navrací první komponent typu T, na který narazí a druhá všechny. Je také možná vybírat komponenty a objekty v rodičovských entitách a nebo všech podřazených entitách.

Součástí třídy GameObject jsou také její statické metody. Nejdůležitější z nich je metoda Find, která zajišťuje hledání objektu ve scéně. Je to ale výpočetně náročná úloha, takže není doporučenu tuto metodu volat často nebo dokonce periodicky.

2.4 UnityEditor

Slouží hlavně jako editor scény, ale také jako nástroj pro správu všech assetů potřebných pro správný běh aplikace. Síla tohoto programu je v tom, že dokáže lehce komunikovat se všemi MonoBehaviour skripty a dokonce zasahovat do jejich práce. Je například velmi snadné propojit objekty, tak aby pomocí skriptů komunikovaly. Všechny veřejné parametry skriptů jsou z editoru lehce dostupné a dají se jednoduchým způsobem nastavit jako výchozí hodnoty bez jakéhokoliv programování.

3 Implementace

Stejně jako většina her tohoto typu je i moje aplikace z velké části jen množstvím triggerů, eventů a periodicky provádějících se činostech (pomocí Unity). Díky tomu, že si Unity umí periodicky volat funkci Update a základní herní smyčka je již vytvořená jsem se mohl soustředit na implementaci skriptů, které by zajistily požadovanou funkci.

Z pomocných prvků Unity používám hlavně fyzikální systém Unity a Input Manager. Fyzikální systém Unity používám z toho důvodu, že úplně vypnout nikdy nejde a nabízí již hotovou kontrolu kolizí a pohyb objektů. V mém projektu fyziku nepotřebuji, ale její používání mi usnadnilo práci. Používám pouze jednoduché posuny po ploše hodně malého množství objektů typu RigidBody2D. Toho docílím přidáním sil k takovým objektům.

Abych zajistil správný vstup od uživatele, používám modul Input Manager. Ten prakticky vzato simuluje fyzický objekt podobný joysticku, který reaguje na uživatelský vstup. Uživatel může použít jako vstupní zařízení klávesnici (pro tu je hra optimalizována), nebo třeba gamepad a joystick. Tento modul sem zawrapoval do vlastní třídy, což mi dovoluje přesnější zpracování vstupu od uživatele.

3.1 Management

V aplikaci používám 3 managery pro správu hry. V následujících kapitolách je krátce popíšu.

3.1.1 Game Manager

Tento manager má na starosti správu průběhu hry od začátku až do konce. Stará se o to kdy hra začíná a končí a také spravuje herní menu. To slouží uživateli k vypnutí nebo restartu hry. Dále má na starosti pauzy a správu ostatních managerů.

Třída, která ho implementuje je návrhového vzoru jedináček. Díky tomu můžu zaručit, že jeden jediný Game Manager bude aktivní po celou dobu hry tzn. i při prohazování scén nebo restartu.

3.1.2 Player Manager

Má na starosti všechny data hráče a také jejich zobrazení na GUI. Stará se o počet životů a také o interakci hráče s inventářem.

Stejně jako v předchozím případě je implementován návrhovým vzorem jedináček. Díky tomu má hráč při přechodu mezi scénami stále stejný počet životů a inventář.

3.1.3 Level Manager

Tato třída se stará o všechny objekty na scéně, které mají svou speciální úlohu. De o NPC, triggery a použitelné objekty. Dála tato třída poskytuje hráči informace o objektech ve scéně, např. ve chvíli když chce hráč zaměřit nepřítele, level manager mu dá informace o všech nepřátelech v jeho okolí.

3.2 Třída Character a její potomci

Všechny postavy ve hře mají společného předka a to je třída Character. Implementuje základní funkce všech postav (jako je například pohyb, útočení, poškození, kontrola kolizí hitboxů atd.).

Pro rozdělení vlastností jednotlivých postav jsem vytvořil několik rozhraní, která třídy dědící Character mohou implementovat. Samotná třída Character implementuje rozhraní IMovable, IAttackable a IScriptable. V projektu je několik dalších rozhraní definujících jiné vlastnosti.

Nejdůležitější třídy dědící Character jsou třídy PlayerCharacterScript, která je postavou hráče a ovládá ji hráč, a třída ZombieMainCharScript, která je zatím jedinou počítačově ovládanou postavou a nepřítelem ve výsledném demu.

4 Uživatelská příručka

Aplikace je vytvořená pro Win64bit systémy. Spouští se pomocí souboru Wolf.exe.

4.1 Obsah archivu

V archivu s testovací verzí aplikace se nachází 4 položky:

1. Složka Wolf_Data
2. player_win_development_x64.pdb
3. Wolf.exe
4. player_win_development_x64_s.pdb

Pokud některý z těchto souborů chybí, nebude možné aplikaci spustit. Poslední položkou v archivu je složka Wolfpack-project, ve které se nachází všechny zdrojové kódy a celý projekt z Unity Editoru.

5 Závěr

Během vývoje jsem bohužel přišel o společníka a nebyl sem schopen najít náhradního grafika. Rozhodl jsem se tedy vytvořit pouze technické demo, na kterém by měly být vidět základní mechaniky hry. Při vývoji se mi podařilo seznámit se s velkou části API Unity a nabral jsem při tom hodně zkušeností ohledně Unity, vytvářen her a práce v týmu.

Na projektu rozhodně chci dále pokračovat a dotáhnout svou vizi do konce. Demo jsem již rozšířil mezi známé a dostal sem většinou kladné ohlasy.