**Střední průmyslová škola a Vyšší odborná škola, Písek,**

**Karla Čapka 402**

**397 11 Písek**

**Školní rok: 2022/2023**

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

**Specializace: Počítačové sítě a programování**

**Zadání maturitní práce**

**3D hra**

Jméno žáka: Pavel Kosík Třída: B4.I

Téma číslo: 13 Vedoucí práce: Mgr. Milan Průdek

Termín odevzdání: 31. březen 2023

Zadaní maturitní práce

1. Vytvořte plně ovladatelnou postavu s přidanými animacemi.
2. Vytvořte vhodné prostředí pro hru.
3. Přidejte prvky hratelnosti napsané pomocí jazyka c++.
4. Navrhněte uživatelské rozhraní se zobrazenými informacemi k hratelnosti (inventář), přidejte menu hry
5. Přidejte hráčem neovládané postavy, se kterými bude možné se setkat (zadávají hráči úkoly)

Anotace

Cílem dlouhodobé maturitní práce spadající pod obor Počítačové sítě a programování bylo vytvořit 3D hru v Unreal Engine 5.

Annotation

The goal of long-graduation work under the Computer network and programming, was to create 3D game in Unreal Engine 5.

Obsah maturitní práce

[Zadaní maturitní práce 2](#_Toc120698960)

[Anotace 3](#_Toc120698961)

[Annotation 3](#_Toc120698962)

[Obsah maturitní práce 4](#_Toc120698963)

[Úvod 5](#_Toc120698964)

[1. Unreal Engine 5 5](#_Toc120698965)

[2. Vytvoření plně ovladatelné postavy 5](#_Toc120698966)

[3. Vytvoření vhodného prostředí 5](#_Toc120698967)

[4. Přidání prvků hratelnosti 6](#_Toc120698968)

[5. Návrh uživatelského rozhraní 6](#_Toc120698969)

[6. Hráčem neovládané postavy 6](#_Toc120698970)

Úvod

Jako dlouhodobou maturitní práci jsem si vybral vytvoření 3D hry v Unreal Engine 5. Téma mě zaujalo, protože jsem si chtěl vyzkoušet práci v Unreal Engine 5.

1. Unreal Engine 5

Unreal Engine 5 je open source vývojové prostředí využívané především pro vývoj počítačových 3D her. Byl vyvinut společností Epic Games. Jedná se již o pátý program řady Unreal Engine. Každá řada měla spoustu verzí, které doplňují dané řady o nové funkce a vylepšují funkce staré. Mezi nové funkce Unreal Engine 5 patří Lumen a Nanite. Funkce Lumen slouží k vykreslování světel v reálném čase. Ve všech minulých řadách Unreal Engine bylo nutné světlo vytvořit a uložit a dané světlo se proto nebylo schopno měnit v reálném čase. Díky využití funkce Lumen není nutné světlo ukládat a možné ho generovat v reálném čase. Tato funkce má samozřejmě vliv na náročnost daného programu. Funkce Nanite slouží k práci s trojúhelníky, z kterých se skládá každý objekt ve hře. Automaticky upravuje velikost a počet trojúhelníků na základě několika faktorů aby bylo dosažené co nejvyšší kvality objektů s co nejnižšími náklady na výkon.



Obrázek

1. Vytvoření plně ovladatelné postavy

Prvním krokem k vytvoření plně ovladatelné postavy je vytvořit si vlastní C++ třídu. Při vytváření nové C++ třídy se Unreal Engine zeptá pomocí, kterého vzoru chceme třídu vytvořit. Každý vzor je něčím specifický a hodí se na něco jiného. Nejvhodnějším vzorem pro hráčovu postavu je třída typu ACharacter. Tuto třídu jsem vybral, protože obsahuje nejvhodnější komponenty pro vytvoření hráčovi postavy. Konkrétně obsahuje komponenty Skeletal Mesh Component, Character Movement Component a Capsule Component. Hlavní funkce Skeletal Mesh Component je vytvoření vhodného modelu hráče. V tomto komponentu stačí vybrat konkrétní mesh, který chceme použít jako výchozí vzhled pro hráčskou postavu. Dále u tohoto komponentu musím nastavit správný způsob jak zpracovávat animace. Pro hráčskou postavu je nejlepší možnost použít takzvaný animační blueprint. Ten nám dovoluje vytvořit jaké konkrétní animace mají proběhnout v určitý moment. Výsledek správně vytvořeného animačního blueprintu si lze představit jako vývojový diagram obsahující spoustu stavů a přechodů mezi nimi. Unreal Engine obsahuje funkci nazvanou Retargeter. Díky této funkce můžu přenést animace z jednoho meshe na jiný mesh. To mi dovoluje si stáhnout všechny potřebné animace z webu, který se jmenuje Mixamo a ty následně importovat do Unreal Enginu. Protože mesh stažený z Mixama má jinou hierarchii kostí než mesh, který jsem si vybral pro hráčovu postavu, musím stažený mesh upravit v 3D editoru. Pro mé účely nejlépe posloužil Blender, protože obsahuje plugin na převedení modelu z Mixama do Unreal Enginu. Po převedení meshe mohu importovat všechny moje animace do Unreal Enginu. Při importaci nesmím zapomenout zakliknout políčko Skeletal Mesh. Kdybych ho nezakliknul tak by Unreal Engine nevytvořil z impotovaných animací mesh, který v Retargeteru použiju pro převedení importovaných animací na hráčův mesh. Pro vhodné nastavení přechodů musím mít proměněné, které správně určují kdy se mohou animace přepnout z jednoho stavu do stavu druhého. Pro lepší organizaci těchto proměnných si vytvořím speciální třídu založenou na bázi {dopln nazev}. Do hlavičkového souboru si vložíme proměnné typu bool pro každý stav, do kterého se může hráč při průběhu hrou dostat. Konkrétně se jedná o stavy {dopln nazev stavu}. Abychom tyto proměnné mohli použít v našem animačním blueprintu musíme u každé z nich určit, že nemá být viditelná pouze v C++ kódu, ale i v Unreal Enginu. To provedeme pomocí funkce UPROPERTY(). Do kulatých závorek této funkce vložím jak chci aby Unreal Engine vidět dané proměnné. Pro proměnné jejichž hodnotu chceme pouze číst a není nutné ji upravovat v Unreal Enginu je nejlepší do kulatých závorek uvést argument BlueprintReadOnly. To mi dovoluje získat v Unreal Enginu přístup k těmto proměnným aniž bych se musel obávat, že dojde k jejich nesprávnému přepsání Unreal Enginem. Další proměnou, kterou bude naše třída starající se o náš animační blueprint potřebovat je reference na třídu hráče, že které získám všechny údaje potřebné k určení stavů hráčské postavy. Pro lepší čitelnost kódu si vytvořím funkci {dopln nazev funkce}. Tato funkce bude volána při každém snímku hry a jejím obsahem je přepsání všech hodnot předem deklarovaných proměnných na jejich aktuální stav. Proto abych předešel případným pádům hry musím volání této funkce vložit do if podmínky. Podmínkou pro volání této funkce je že se povedlo naší třídě získat referencí na hráčovu třídu. Bez této reference by mohlo dojít k tomu, že bych se snažil získávat data z neexistujícího místa v paměti a to by způsobilo pád programu. Obsahem funkce je dosazení konkrétních hodnot načtených z hráčovi třídy. Další komponent je CharacterMovementComponent. Tento komponent obsahuje všechny nastavení týkající se pohybu hráče. Využil jsem ho pro snazší implementaci různých rychlostí chůze. Tento komponent totiž obsahuje vlastní funkci pro pohyb. Další věc, co musíme udělat je vybrat vhodný model pro hráčovu postavu.

1. Vytvoření vhodného prostředí

S tvorbou vhodného prostředí nám pomůže plugin v Unreal Engine 5. Jedná se o plugin {dopln jmeno pluginu}. Tento plugin nám dovoluje pomocí nastavení několika proměnných vytvořit realisticky vypadající tvar připomínající ostrov. Tento tvar můžeme použít jako základ, na kterém budeme dále stavět. Další krokem je vložit na tento ostrov vhodné textury. Tyto textuty získáme na Unreal Engine Asset Store. Po jejich aplikaci na vytvořený tvar ostrovu začne tento tvar vypadat jako opravdový ostrov. Další krokem je osadit ostrov přírodními prvky, které by byly k nalezení na ostrově. V našem případě se jedná o různé stromy, horniny, keře, květiny a další. Všechny tyto přírodní prvky získáme z Unreal Engine 5 Asset Store. Abychom si ušetřili čas a nemuseli, každý jednotlivý přírodní prvek umísťovat jednotlivě můžeme, využít nástroje {dopln presny nazev nastroje}. Tento nástroj nám dovolí rozmístit více prvků najednou pomocí zadaných parametrů. Jako například hustota prvků, jejich velikost a tak dále.

1. Přidání prvků hratelnosti

Prvky hratelnosti jsou prvky, které obohacují hratelnost abych si hráč hru více užil. Nejdůležitějším prvkem hratelnosti je umožnit hráčem ovládané postavě ovlivnit ostatní neovládané postavy ve vytvořeném světě. Hráč je díky tomu schopen lépe se ponořit do herního světa a má pocit, že jeho rozhodnutí mají smysl a hru jsou schopné ovlivnit.

* 1. **Možnost hráče zabít nepřátelské neovládané postavy**

Jeden z hlavních prvků hratelnosti je možnost zabít nepřátelské postavy. Tento prvek je naprosto stěžejním prvkem hry, protože hráč díky němu může překonat překážku, kterou tyto nepřátelské postavy představují. Při zabití nepřátelské postavy může dojít k tomu, že z nepřítele vypadne odměna, kterou může hráč sebrat. Touto odměnou můžou být různé zbraně, výzbroj a tak dále. Aby hráč mohl zabít nepřítele musí mít nepřítel takzvaný hitbox. Jedná se o oblast, kterou pokud se hráčovi povede zasáhnout útokem, tak dojde k udělení poškození nepříteli. Tento hitbox zprostředkovává CapsuleComponent. Tento komponent má tvar kapsule, které lze nastavit šířku a výšku. Pro hitbox nepřítele je vhodné vytvořit kapsuli, která je o něco vyšší a širší, než je model nepřítele. Díky tomu nikdy nedojde k tomu, že by hráč vizuálně zasáhl nepřítele, ale nepřítel nedostal poškození. Proto aby nepřítel mohl být zraněn musí mít číselnou hodnotu, která bude symbolizovat jeho životy. Životy budou symbolizované číslem s desetinou čárkou, takzvaným floatem. Při každém zasažení nepřítele hráčem toto číslo snížíme o hodnotu hráčova útoku. Pokud životy nepřítele poklesnou na nulu nebo níž. Spustí se animace smrti a nepřítel je vypnut. Z předem daných nepřátel může vypadnout odměna.

1. **Možnost hráče dostat úkoly od přátelských neovládaných postav**

Dalším prvkem hratelnosti je možnost hráče dostat úkol od přátelských neovládaných postav. Tento prvek je zde pro to, aby měl hráč ve hře co dělat. Ovšem ne každá přátelská hráčem neovládaná postava může dát hráči úkol. To můžou pouze předem zvolené postavy, u kterých to dává logický smysl. Hráč si úkol vyzvedne rozhovorem s postavou, která úkol zadává. Při rozhovoru s postavou hráč vyslechne, co se od něj očekává. Hráč se může následně rozhodnou, zda chce daný úkol splnit nebo ne. Pokud se rozhodne, že se pokusí úkol splnit tak bude úkol přidán do záznamu úkolů. Hráč zde vidí zadání úkolu a jeho současný postup úkolem. Po splnění úkolu si může hráč vyzvednout odměnu za provedení úkolu od postava co daný úkol zadala.

1. Návrh uživatelského rozhraní

Uživatelské rozhraní je nedílnou částí hry. Jedná se o různá menu a tak podobně. Slouží aby hráč mohl měnit nastavení, spouštět hru a další různé ukazatele a tak podobně.

1. **Úvodní uživatelské rozhraní**

Toto uživatelské rozhraní vidí hráč po spuštění hry. Obsahuje celkem tři tlačítka. Tlačítko pro zapnutí levelu, tlačítko pro otevření nastavení hry a tlačítko pro vypnutí hry. Po stisknutí tlačítka pro zapnutí levelu se hráči zobrazí menu, kde má možnost vybrat si zda chce načíst jednu ze svých uložených pozic ve hře. Nebo začít hrát od znovu.

1. **Inventář**

Toto uživatelské rozhraní se stará o všechny položky, které se hráč při průběhu hry rozhodne sebrat. Obsahuje tedy různé zbraně, výzbroj, lektvary a tak podobně. Má dvě části. Jednu, kde jsou obsaženy všechny hráčovi předměty. A další část, kde si hráč může vybrat konkrétní vybavení které chce v danou chvíli používat. Vybavení, které hráč chce používat může dostat z jedné části do druhé buď přetažením daného předmětu nebo dvojitým kliknutím na obrázek daného předmětu v první části. K vytvoření inventáře použijeme Blueprinty. Tato funkce Unreal Enginu nám dovoluje vytvořit chování uživatelského rozhraní velmi rychle a stabilně aniž bychom museli psát kód v c++. Tento způsob je lepší z důvodu velice špatné podpory a implementace uživatelského rozhraní v c++.

1. **Zdraví hráče**

Stejně jako nepřátelské postavy má i hráč životy. Pokud by je neměl tak by nemohl být zabit nepřátelskými postavami a hra by postrádala smysl a nebyla by moc zábavná.

1. Hráčem neovládané postavy

Hra se neobejde bez hráčem neovládaných postav. Ty se rozdělují v tomto případě na dva druhy. A to je postavy, které jsou vůči hráčovi přátelské a nepřátelské. Těchto postav je ve hře více druhů, ale mají některé společné vlastnosti. Jednou z těchto vlastností je pohyb po okolí. Pohyb po okolí je realizován pomocí komponentu v Unreal Engine 5, který se nazývá Nav Mesh. Pro správné fungování Nav Mesh je potřeba do levelu vložit Nav Mesh Bounds Volume. To vede k vytvoření plochy, po které se můžou neovládané postavy pohybovat. Pro správnou funkci této Nav Mesh je nutné v RecastNavMesh-Default zapnout dynamickou generaci. To vede k vytvoření nové Nav Mesh při každé změně týkající se plochy, po které se neovládané postavy pohybují. Takovou změnou může třeba být přidání překážky přes, kterou se nemohou dostat a tak podobně.

1. **Nepřátelské postavy**

Tyto postavy slouží jako překážka, kterou musí hráč překonat, aby splnil různé úkoly. Ve hře je jich více druhů, z nichž se každá chová jinak.

1. Kostlivec

Prvním nepřítelem je kostlivec. Tento nepřítel je specifický tím, že necítí bolest a tudíž po obdržení poškození od hráče mu okamžitě vrátí úder. Hráč tudíž musí rychle reagovat a vyhnout se útoku aby nebyl nepřítelem zabit.

1. Závěr
2. Zdroje

https://www.unrealengine.com/en-US/unreal-engine-5