Online multiplayer bojová hra

Dokumentace maturitní práce

Jiří Bittner

školní rok: 2023/2024 třída: 8. V

Zadání maturitní práce:

Student v herním enginu Unity vytvoří bojovou plošinovou hru pro více hráčů za pomoci knihovny Photon 2. Inspirací je série her Super Smash Bros. a Brawlhalla. Hra navíc bude rozšířena o možnost vytvoření vlastních předmětů do hry - např. postav, zbraní nebo i samotného levelu.

Teoretická část - Serializace dat v síťových hrách

Prohlašuji, že jsem na práci pracoval samostatně pouze za pomoci použitých zdrojů a že v práci i v dokumentaci jasně vymezuji, které části kódů jsou mým originálním dílem, které jsou upravenou verzí a které jsou převzaty v plném rozsahu.

---

podpis žáka

Obsah

1. TEORETICKÁ ČÁST

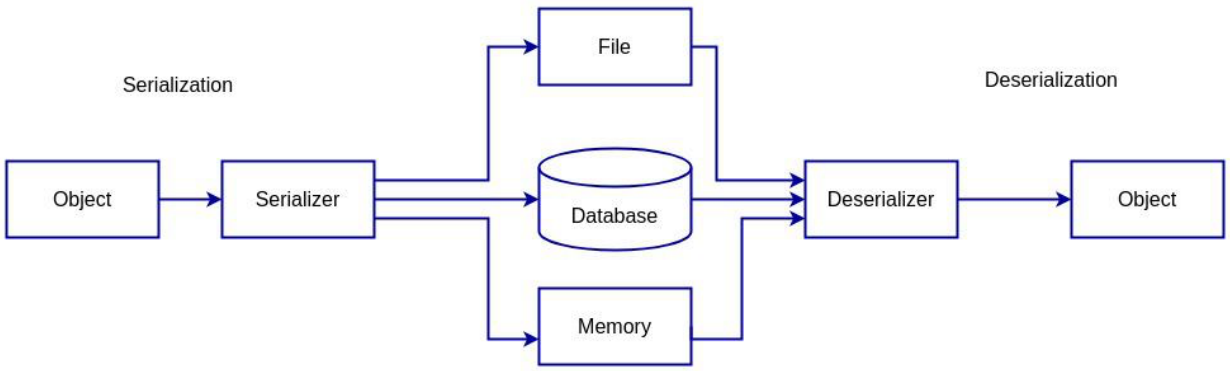
Tato kapitola prezentuje teoretickou část maturitní práce týkající se serializace dat v síťových hrách. Téma serializace jsem vybral, protože je nedílnou součástí síťové komunikace, kterou ve svém projektu také využívám.

1.1. Serializace a deserializace

Serializace je proces přeměny vybrané datové struktury na objekt, který je možné uložit nebo poslat přes internet a poté ho zrekonstruovat jako klon původní datové struktury. První serializační metody vznikly již v 60. letech 20. století vytvořením jazyku GML (Generilized Markup Language), ze kterého se později vyvinul jazyk XML. V případě jednoduchých struktur se může využít serializace do textové podoby (například do formátu JSON, XML), u rozsáhlejších dat se využívá serializace do binární podoby pro úsporu objemu dat. Objekt, který provádí serializaci se nazývá serializátor.

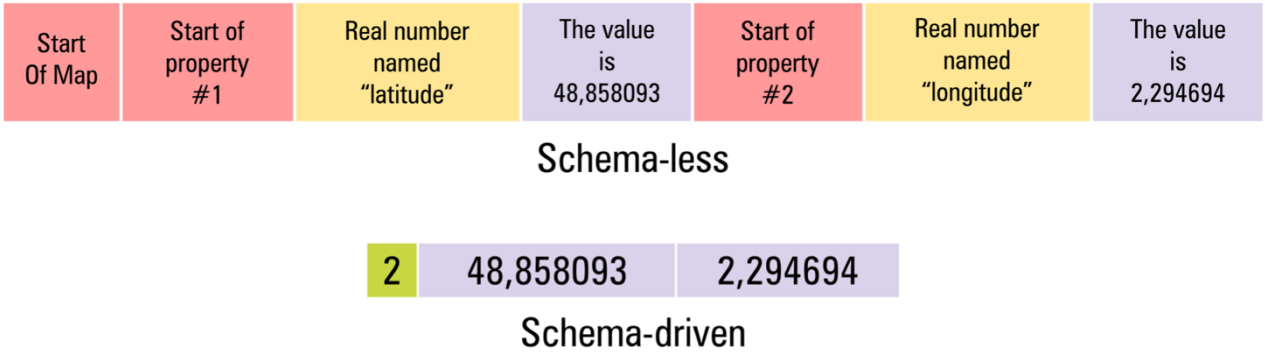
Deserializace je proces opačný k serializaci. Z binárních dat nebo textového souboru se znovu vytvoří původní datová struktura.

Serializace a deserializace umožňují přenos dat, který není závislý na konkrétním použitém hardwaru a programovacím jazyce. Na obrázku 1 můžeme vidět proces serializace, přenosu dat a deserializace.



Obrázek 1. Serializace a deserializace. Převzato z online článku Baeldung [1].

Rozlišujeme serilalizaci se schématem a bez schématu. Serializace se schématem využívá předem známou strukturu dat (schéma), přičemž popis této struktury není obsažen ve výsledném serializovaném souboru. Tento způsob serializace se využívá většinou pokud jsme autory serializačního i deserializačního kódu a tím pádem víme, v jakém pořadí jsou data zapsána. Serializace bez schématu obsahuje popis serializované hodnoty společně s hodnotou samotnou. Výhodou je, že nemusíme znát pořadí hodnot, nevýhodou je větší množství dat v serializovaném souboru. Na obrázku 2 lze vidět rozdíl mezi serializací se schématem a bez schématu.



Obrázek 2. Rozdíl mezi schématickou a neschematickou serializací, převzato z článku Viottiho a Kinderkhedia [2].

1.2. Serializace do textové podoby

Výhodou serializace do textového souboru je možnost vidět jeho obsah a upravovat ho. Nevýhodou je větší velikost serializovaného souboru a jeho delší načítání. Nejrozšířenější formáty jsou JSON (JavaScript Object Notation) nebo XML (eXtensible Markup Language). Proces serializace do JSON souboru spočívá v převedení požadované třídy do textové podoby, kde budou zachovány všechny vazby, pojmenování a hierarchie dat, jedná se tedy o serializaci bez schématu. Příklad serializace třídy „Auto“ do souboru JSON je uveden na obrázku 3.

Objekt v C#	Serializovaná JSON data
Class Auto { *String name = "Skoda";* *int numberOfSeats = 5;* *String passengers =* *["Honza","Jirka","Jakub"]* *}*	*{"name": "Skoda","numberOfSeats": 5,"passengers":* *["Honza","Jirka","Jakub"]}*

Obrázek 3. Příklad serializace do JSON formátu.

1.3. Serializace do binární podoby

Binární serializace přemění datovou strukturu na proud bitů (1 nebo 0). Používá se většinou v kombinaci se serializací se schématem, pro dosažení menší velikosti serializovaných dat a tím i kratší doby přenosu. Při binární serializaci je potřeba správně ošetřit ukládání a načítání základních datových typů, jejichž formát se může na různých architekturách lišit (Big Endian, Little Endian) [2].

1.4. Photon 2 serializace

Photon 2 je veřejně dostupná knihovna pro herní engine Unity, která umožňuje síťovou komunikaci. V Unity nabízí hned několik možností serializace dat a jejich následného přenosu přes síť ostatním hráčům. Photon podporuje serializaci základních datových struktur jako je například integer, float, string, double a pole datových struktur. Pro složitější struktury se dá vytvořit vlastní metoda, která serializuje a deserializuje daný objekt, tato metoda se pak dá registrovat u Photonu pro použití v dané aplikaci. Ve svém kódu využivám metody RPC, RaiseEvent a OnPhotonSerializeView, všechny tyto metody využívaji serializaci od Photonu.

2. CÍLE PRÁCE

Mým cílem bylo vytvořit online bojovou hru pro více hráčů. Inspiroval jsem se tituly jako jsou Super Smash Bros. a Brawhalla. Zároveň jsem chtěl do své hry přidat něco navíc, co by tu mojí odlišilo. Rozhodl jsem se vytvořit systém, kde si hráč může nakreslit své vlastní zbraně, postavu, a dokonce i herní plochu. Tyto nakreslené objekty se poté přenesou ke všem hráčům, kteří se rozhodnou přidat se do hry.

3. ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

Program by se dal rozdělit na tři hlavní části: vytváření herních objektů, síťová komunikace s ostatními hráči a samotná logika hry. Tyto části podrobněji popisuji v kapitolách 3.1., 3.2. a 3.3.

3.1. Vytváření herních objektů

Hráč má možnost nakreslit své vlastní předměty do hry. Může nakreslit herní mapu, postavu, meč, sekeru, kopí, pistoli, kulku a bombu. K dispozici má tužku a gumu. Hráč může měnit průměr těchto nástrojů a pokud se rozhodne, může vše vymazat a začít od začátku. Kreslí se do čtvercové mřížky zvětšených pixelů, každý tento pixel je jeden herní objekt. Na obrázku 4 je vidět přehled vytvořených předmětů.

Obrázek 4. Přehled vytvořených předmětů.

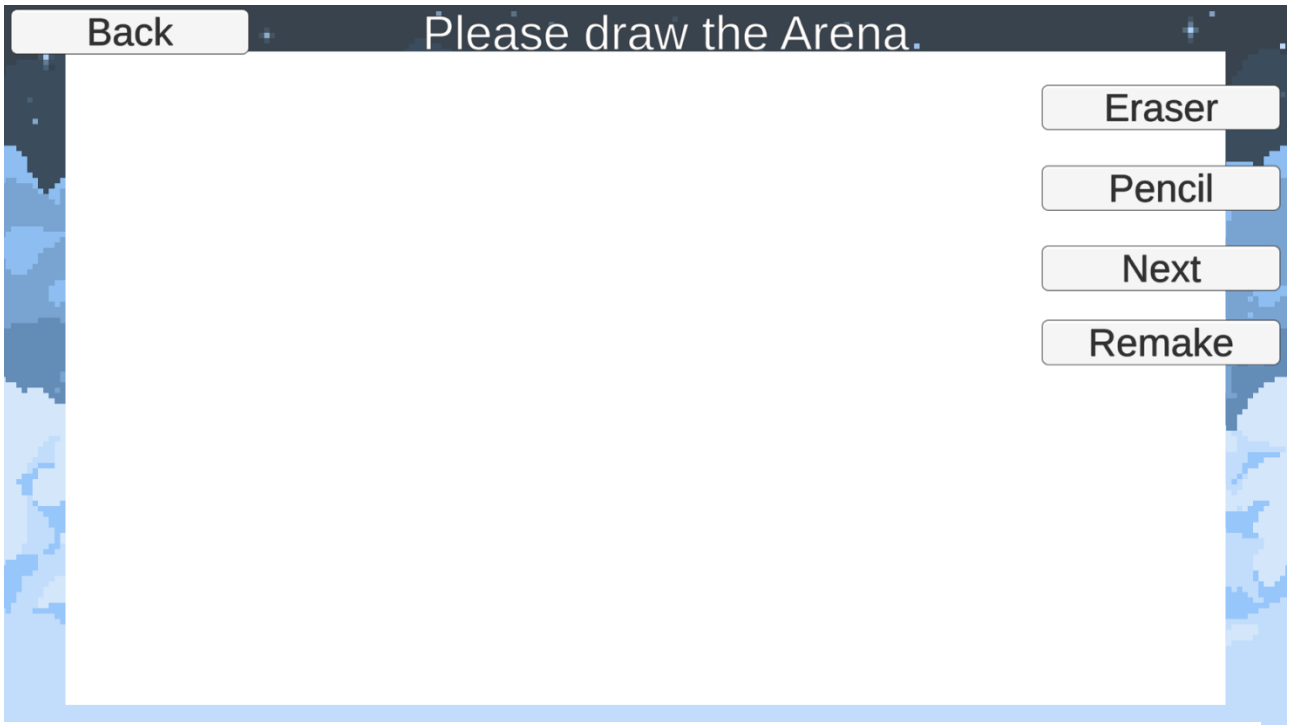
3.1.1. Kreslení předmětů

AnalyzeImage je skript, který řeší veškerou logiku, týkající se vytváření předmětů. Tento skript obsahuje dvě třídy: Prefab a Pixel. AnalyzeImage je přiřazený hernímu objektu s názvem Controller ve scéně CreatePrefabs.

Úkolem třídy Prefab je tvorba textury z nakresleného objektu a následná manipulace s herním objektem, na který se tato textura připojí. Tento objekt slouží jen jako vizuální reprezentace finálního herního objektu a ještě není plnohodnotným objektem, který se bude používat ve hře.

Třída Pixel vytvoří herní objekt, který reprezentuje jeden pixel v malovacím poli. Tato třída má metody, díky kterým můžeme přebarvit daný pixel a zároveň uchovává aktuální barvu pixelu pro vytvoření textury ve třídě Prefab.

Na začátku se zavolá funkce, která vytvoří dvojrozměrné pole tříd Pixel a tím vytvoří i malovací plátno, na které může hráč malovat. Poté se zobrazí vyskakovací okno upozorňující hráče, jak má správně kreslit objekty. Poté, co hráč vyskočí ze zmíněného okna, se zapne funkce umístěná v metodě update, která vysílá paprsek od pozice kamery do pozice myši a kontroluje, zda paprsek zasáhne nějaký objekt. Pokud paprsek zasáhne objekt a zároveň hráč mačká levé tlačítko myši, vybrané zasažené objekty se upraví podle zvoleného nástroje (tužka pixely zabarví, guma je naopak odbarví). Může se však stát, že hráč pohybuje s myší moc rychle a funkce update nestíhá změnit všechny objekty. Vytvořil jsem proto funkci, která na základě minulé pozice a současné pozice dopočítá, které pixely by se měly dodatečně změnit. Toho je dosaženo vysláním paprsku z minulé pozice do současné pozice a určením, které objekty kolidují s tímto paprskem. Obrázek 5 ukazuje uživatelské rozhraní pro kreslení předmětů.



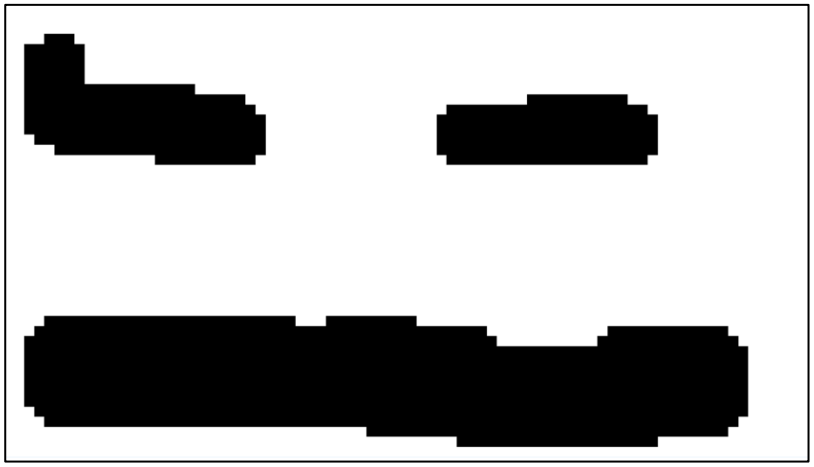
Obrázek 5. Prostředí pro tvorbu předmětů.

Poté, co hráč nakreslí všechny předměty, se pro každý z nich vytvoří herní objekt, pro který se vytvoří textura zobrazující nakreslený předmět. Textura se vytváří na základě dvojrozměrného pole celých čísel, které se vytvoří z pole tříd Pixel (pokud je pixel zabarvený pak je to 1, pokud není zabarvený pak je to 0). Zvolí se šířka pixelu pro texturu a poté se prochází polem celých čísel. Pokud se narazí na 1, tak se daná pozici v textuře zabarví. Nakonec se zavolá funkce, která zobrazí všechny vytvořené objekty pro kontrolu. Pokud hráč klikne na zobrazený předmět, vrátí se do kreslícího prostředí a může ještě objekt upravovat. Po úpravě se vytvoří nová textura stejným způsobem.

Když je hráč spokojen se svými předměty, může kliknout na tlačítko Start Game. Po kliknutí se hráči zobrazí vyskakovací okno umožňující uložit vytvořené objekty. Po ukončení tohoto okna se uloží dvojrozměrné pole celých čísel reprezentující každý vytvořený objekt do herního objektu s názvem AssetHolder. Tento objekt se přenese do další herní scény, kde pomůže vytvořit finální verze těchto herních objektů.

3.1.2. Ukládání vytvořených předmětů

Hráč se může rozhodnout, zda chce své předměty uložit pro použití v dalších hrách. Pokud se rozhodne je uložit, dvojrozměrné pole celých čísel se převede na jednorozměrné pole. Toto pole dále dělíme na skupiny celých čísel po čtyřech. Protože pole obsahuje jen 1 a 0, každou tuto skupinu můžeme vyjádřit jako jeden hexadecimální znak. Tyto znaky se ukládají do řetězce a ten se poté uloží společně s řetězci ostatních vytvořených objektů do nově vytvořeného textového souboru. Tento soubor poté uložíme do aplikačního adresáře hry s koncovkou .brawlGame. Soubor poté jde otevřít v textovém editoru pro kontrolu dat a jejich případnou úpravu. V budoucnu by bylo možné projekt rozšířit o ukládání těchto dat v binární podobě. Na obrázku 6 je vidět textová podoba jednoho herního předmětu.



00000000000000000000000000007FFFF8000000000000FFFFFFFFF000FFFFFFFFFFFFFFFFF801FFFFFFF
FFFFFFFFFFC03FFFFFFFFFFFFFFFFFC03FFFFFFFFFFFFFFFFFC03FFFFFFFFFFFFFFFFFC03FFFFFFF
FFFFFFFFFFC03FFFFFFFFFFFFFFFFFC03FFFFFFFFFFFFFFFFFC03FFFFFFFFFF801FFF801FFFFFFFF
FFF000FFF000FFFFFF8FF8000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000001FF80000FFFFF000007FFFFC0001FFFFF80001FFFF
FC0001FFFFF80003FFFFFC0001FFFFF80003FFFFFC0001FFFFF80003FFFFF80000FFFFF00003FFFFF00000
00FFC00003FFFE000000000000003F000000000000000003F000000000000000003F000000000000000003F000
00000000000000E000000000000000000000000000000000000000000000000000000000000000000000000000

Obrázek 6. Herní předmět společně s jeho textovou reprezentací.

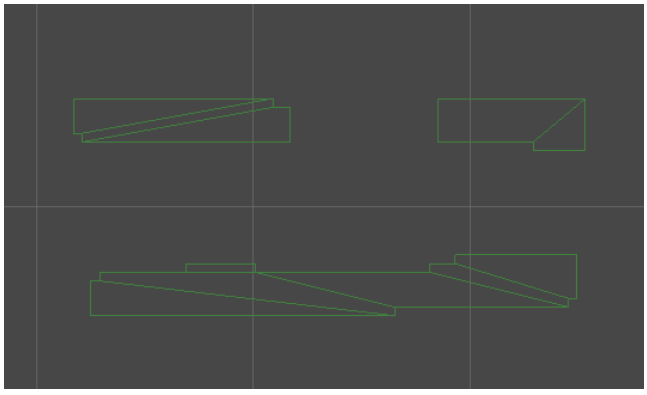
3.1.3. Načítání vytvořených předmětů

V případě, že se hráč rozhodne vybrat si z již vytvořených předmětů, skript FetchCreatedLevels se pokusí najít všechny soubory v aplikačním adresáři hry s koncovkou .brawlGame. Pokud nějaký soubor najde, zobrazí pro něj tlačítko, díky kterému může hráč načíst tyto předměty. Jestliže hráč klikne na toto tlačítko, načteme scénu LoadSavedAssets. Tato scéna je stejná jako scéna, ve které se vytváří nové předměty s tím rozdílem, že díky tomu, že už máme načtené předměty, můžeme rovnou přeskočit do části, kde je zobrazíme pro kontrolu.

3.1.4. Vytvoření finálních objektů

CreatePrefab je script umístěný ve scéně GameLevel. Poté co se scéna načte, si tento script najde herní objekt s názvem AssetHolder, ze kterého získá dvojrozměrné pole pro každý předmět. Poté se pro každý předmět aktivuje funkce, která zavolá síťovou funkci Photonu, RaiseEvent. Funkce RaiseEvent vytvoří novou událost v aktuální místnosti, které přiřadíme serializované pole celých čísel. RaiseEvent se volá, protože chceme vytvořit herní objekty u všech hráčů, kteří se rozhodnou připojit se do této hry. Pro zpracování přijatých informací musíme v kódu implementovat funkci onEvent, která se zavolá u všech hráčů pokaždé, když se vytvoří nová událost. V této funkci kontrolujeme, zda se jedná o správnou událost a pokud ano, tak zavoláme funkci CopyComponents, která vytvoří finální objekt.

CopyComponents vezme předpřipravený odpovídající herní objekt, který je ve složce Resources a přidá mu texturu, která se vytvoří stejným způsobem jako v AnalyzeImage. Pro finální herní objekt však se musí vytvořit i PolygonCollider2D. Bohužel automatické vytváření tohoto kolizního tělesa na vytvořenou texturu dělalo problémy. Vytvořil jsem proto funkci CreatePolygonCollider, která má za úkol na základě souřadnic zabarvených pixelů vytvořit nové kolizní těleso, které odpovídá tvaru textury. Toho se dosáhne obcházením tělesa po obvodu, takto vytvořené těleso je vidět na obrázku 7.

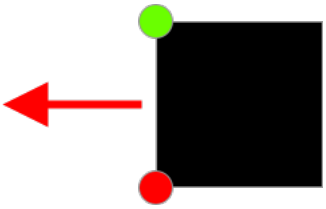


Obrázek 7. Vytvořené kolizní těleso.

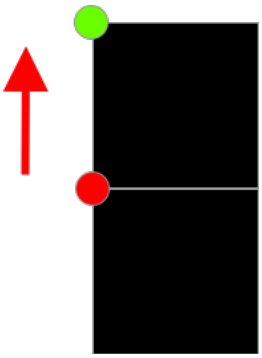
Nejdříve se najde nejdolnější levý zabarvený pixel. Zvolíme počáteční roh kolizního tělesa jako levý dolní roh tohoto pixelu a nastavíme směr pohybu jako levý jednotkový vektor (obrázek 8). Musíme si také zapamatovat na kterém pixelu zrovna stojíme. Při běhu algoritmu existují čtyři možnosti jak postupovat dále v závislosti na tom, s kolika dalšími zabarvenými pixely náš roh sousedí.

Obrázek 8. Herní aréna s červeně vyznačeným prvním rohem.

První variantou je, že roh sousedí jen s jedním zabarveným pixelem, v tom případě vybereme roh, který je osově souměrný s aktuálním rohem podle osy směru pohybu. Tato varianta je vyobrazena na obrázku 9, červená barva reprezentuje aktuální roh, zelená barva zobrazuje roh, na který se bude postupovat. Nový směr pohybu je vždy vektor z předchozího vrcholu do aktuálního vrcholu, to platí u všech čtyř možností. Tato varianta se vždy zavolá po spuštění algoritmu, protože první roh sousedí jen s jedním pixelem. Druhou variantou je, že aktuální roh sousedí se dvěma zabarvenými pixely, v tomto případě pokračujeme na nový roh, který najdeme posunem z aktuálního rohu ve směru pohybu. Tato situace je vyobrazena na obrázku 10.

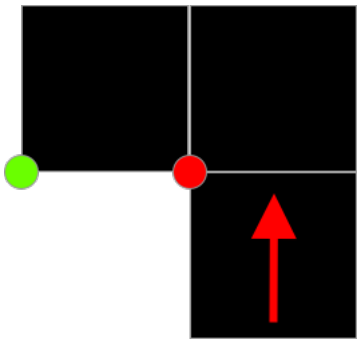


Obrázek 9. Dotyk s jedním pixelem.

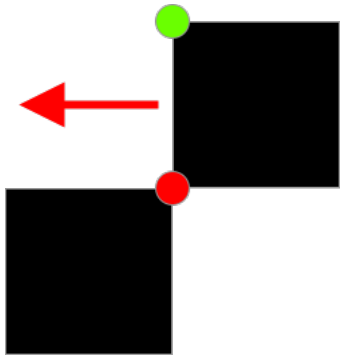


Obrázek 10. Dotyk s dvěma pixely.

Třetí možnost je, že roh sousedí s třemi pixely. V takovém případě se bude postupovat na pixel, který není ve směru pohybu z aktuálního pixelu (obrázek 11). Poslední možností je, že aktuální roh sousedí s dvěma pixely, ale mají dotyk přes aktuální roh (jsou středově souměrné). V tomto případě se algoritmus zachová stejně jako v prvním případě, tedy jako když roh sousedí jen s aktuálním pixelem (obrázek 12).



Obrázek 11. Dotyk s třemi pixely.



Obrázek 12. Rohový dotyk dvou pixelů.

3.2. Síťová komunikace

Photon 2 nabízí pro komunikaci přes internet již zmíněnou funkci RaiseEvent, v kódu však využívám i jiné způsoby komunikace. Mezi ně patří metody RPC (Remote Procedure Call) a OnPhotonSerializeView. Photon také nabízí metodu Instantiate, která nahrazuje metodu Instantiate od Unity. Díky této metodě jsme schopni vytvořit objekt se stejným ID u všech hráčů. Této metodě se předá jméno objektu a ona vytvoří novou instanci daného objektu. Tento objekt však musí být ve složce Resources, anebo v našem případě musí být ve slovníku PhotonDefaultPool, kam se mohou přidávat objekty i za běhu hry.

3.2.1. RPC

RPC metody dokáží zavolat zvolenou funkci na místním i vzdáleném zařízení. K metodě můžeme přidat jako parametry hodnoty, které je Photon schopen serializovat. Můžeme si vybrat, u koho se RPC metoda zavolá a dokonce můžeme zvolit, zda se má metoda zavolat i nově příchozím hráčům pomocí atributu buffered. RPC se dá volat jen na objektu který má komponentu PhotonView a pokud tento objekt odstraníme, odstraníme tím i všechny RPC metody na něm zavolané. V mé hře se RPC metody využívají pokaždé, když je potřeba nějakou funkci zavolat i pro ostatní hráče. RPC se tedy volá na začátku i konci hry, pokaždé když nějaký hráč vezme nebo pustí zbraň, pro útok zbraní a animace zbraní. Volá se také pro změnu jména hráčovy postavy na jeho Photon ID (unikátní identifikátor každého hráče v místnosti), abychom poté mohli najít, jakému hráči patří jaká postava.

3.2.2. OnPhotonSerializeView

Metoda OnPhotonSerializeView se využívá pro změnu pozice herních objektů u ostatních hráčů. Tato metoda implementuje třídu PhotonStream, na které můžeme zavolat funkci SendNext, která serializuje danou hodnotu a pošle ji přes síť ostatním hráčům, kteří ji mohou přijmout funkcí RecieveNext. Takto můžeme posílat aktuální pozici a rotaci našeho hráče a nastavit tuto hodnotu reprezentaci naší herní postavy u ostatních hráčů. Tímto se synchronizují instance herních postav a zbraní, takže u všech hráčů budou na stejném místě. Na rozdíl od funkce RPC se OnPhotonSerializeView volá automaticky a pravidelně, zatímco RPC se zavolá jen jednou.

3.3. Herní logika

Hned po zapnutí hry se objeví menu, ze kterého máme tři možnosti dalšího postupu. Hráč může vytvořit nové herní předměty, jak to bylo popsáno v kapitole 3.1., nebo může otevřít jeho uložené předměty. Poslední možností je připojit se ke hře vytvořené jiným hráčem. Pro tuto možnost potřebuje ID místnosti, do které se chce připojit. Ať už si hráč vybere jakoukoliv z možností, nakonec se dostane do scény GameLevel, kde se odehrává samotná hra.

3.3.1. Přípravy před hrou

Ještě než může hra začít, musí se najít místa, kam se mohou umisťovat hráči a zbraně. Tato místa se najdou pomocí vytvořeného kolizního tělesa herního světa. Pro každou souvislou část tohoto tělesa (herní svět může mít více částí) se najde jeho nejmenší a největší souřadnice x, zároveň se zjistí jeho nejvyšší souřadnice y. Když známe tyto tři body, můžeme procházet body od nejmenšího x po největší x se souřadnicí y o něco větší než tou, kterou jsme nalezli. Kolem každého takového bodu se vykreslí kruh o průměru herní postavy, aby se zjistilo, zda paprsek zasáhne nějakou část herního světa. Pokud žádnou část nezasáhne, může se tento bod přidat na seznam míst, kam se mohou umisťovat herní předměty.

3.3.2. Začátek hry

Poté, co se hráč, který vytvořil místnost, rozhodne zahájit hru, spustí se odpočet herního času a všem hráčům se zapne možnost se pohybovat. Od této chvíle se také začnou ve hře objevovat zbraně, které se vybírají náhodně a umisťují se na náhodná místa ze seznamu míst, který se vytvořil před hrou.

3.3.3. Pohyb hráče

Hráč se může pohybovat doleva, doprava a skákat. Postava se do stran pohybuje na základě ovládání reálným hráčem. Pokaždé, když změníme směr pohybu, se postava musí otočit o 180 stupňů. Jsou k dispozici celkem dva skoky, které se resetují pokaždé, když se postava vrátí na zem. Hráč má navíc možnost sprintovat, pokud zmáčkne klávesu Shift. To hráči poskytne zrychlený pohyb na krátkou dobu. Po použití se sprint musí znovu nabít, což trvá tři sekundy.

3.3.4. Přiřazování zbraní

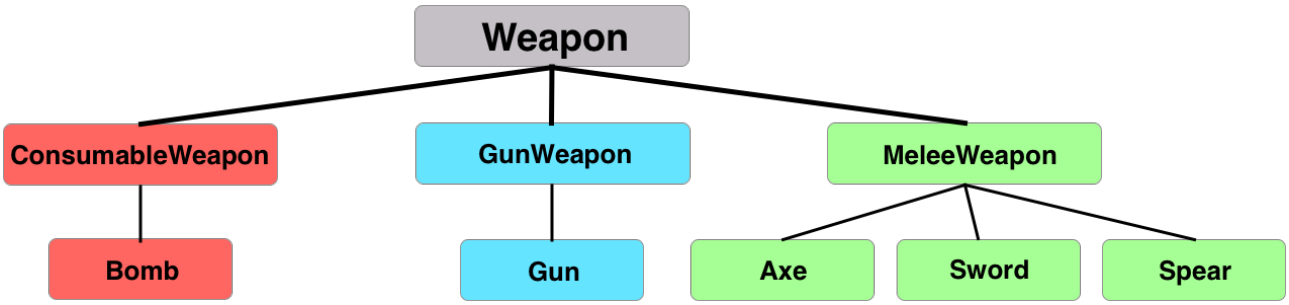
Aby hráč mohl útočit na ostatní hráče, musí mít v ruce zbraň. Tu může získat tak, že se k ní přiblíží a zmáčkne klávesu E. Každá zbraň má své vlastní kolizní těleso, díky kterému se kontroluje, kdy náš hráč přijde do styku s tímto tělesem. Pokud se tak stane, zkontroluje se, zda hráč již nějakou zbraň nedrží a jestliže zmáčkne klávesu E, přiřadí se hráči tato zbraň a on je schopen ji nyní používat. Zároveň se musí zbrani přiřadit naše herní postava jako rodičovský uzel v hierarchii scény, tím se dosáhne, že se zbraň bude pohybovat s naším hráčem.

Pouštění zbraní funguje podobným způsobem. Když hráč zmáčkne E, rodičovský uzel se nastaví na prázdnou hodnotu a tím se zbraň přestane pohybovat s hráčem. U hráče se nastaví hodnota, která drží informaci o tom, zda držíme zbraň, na nepravdu. Nakonec nastavíme animaci zbraně na nečinnou, tím se zbraň začne pohybovat lehce nahoru a dolů. Ostatní hráči tím dostanou najevo, že tato zbraň je znovu volná.

3.3.5. Zbraně

Pro všechny zbraně existuje abstraktní třída s názvem Weapon, která obsahuje abstraktní metodu pro útok. Abstraktní třídy nutí všechny další třídy, které od ní dědí, implementovat její abstraktní metody. Každá zbraň má svou vlastní třídu, která dědí třídu Weapon. Zde je specifická implementace funkce pro útok. Tímto řešením jsme schopni v třídě Weapon zavolat metodu pro útok a pro zvolenou zbraň se zavolá její správná metoda pro útok, která definuje původní zavolanou metodu. Tento systém je implementovaný proto, aby se mohl použít jeden univerzální kód pro útočení zbraní hráčem nehledě na to, kterou zbraň drží v ruce.

Zbraně se dále dělí na blízké zbraně, jednou použitelné zbraně a pistole. Každá tato podtřída má svojí implementaci metody pro odhazování nepřítele. Každá zbraň má svou metodu pro výpočet vektoru odpalu, který se počítá na základě pozice nepřítele a zbraně v moment, kdy ho zbraň zasáhla. Tento vektor se pak předá již zmíněné metodě pro odhazování nepřítele, která díky němu, síle odpalu a procentu opotřebení hráče správně odhodí nepřítele. Na obrázku 13 je vidět hierarchie dědičnosti třídy Weapon.



Obrázek 13. Hierarchie dědičnosti třídy Weapon.

3.3.6. Herní skóre

Jakmile je nějaký hráč odpálen z herního pole, narazí do kolizního předmětu, který obepíná celé herní pole. Tento objekt zavolá funkci, která hráče znovu vrátí do hry na náhodně vybranou pozici. Zároveň tato funkce odebere hráči jeden bod a resetuje jeho procento opotřebení, které přidává na síle útoků protihráčů. Pokud nějaký nepřítel hráče předtím než spadl z mapy odpálil, přidá se tomuto protihráči bod za poražení zmíněného hráče.

3.3.7. Uživatelské prostředí hry

Na začátku hry se zakládajícímu hráči zobrazí tlačítko pro začátek hry, ostatní hráči vidí text, který jim říká, že se čeká na zakládajícího hráče, který musí zahájit hru. Na obrazovce jsou dvě tlačítka, která vidí všichni hráči. Jedno slouží pro opuštění hry a druhé otevře nápovědu pro ovládání hry. Každý hráč má navíc na dolní straně hry plaketu s ikonou svého hráče, jeho názvem, skórem a procentem opotřebení (obrázek 14).

Obrázek 14. Plaketa hráče.

Po vypršení herního času se hra automaticky ukončí a hráčům se zobrazí výsledky podle jejich dosaženého skóre. Uživatel se nyní může rozhodnout, zda chce hrát znovu nebo odejít. Jakkoliv se rozhodne, jeho rozhodnutí se zobrazí ostatním hráčům barevným indikátorem u jeho výsledku. Novou hru může zahájit zase jen originální hráč. Pokud se však tento hráč rozhodne odejít, přenese se tato odpovědnost na jiného hráče.

4. PROGRAMOVACÍ PROSTŘEDKY

Hra byla vytvořena ve vývojovém prostředí Unity, které využívá k programování jazyk C#. Dále je využita knihovna Photon Unity Networking 2, která je volně dostupná v Unity Asset Store. Tuto knihovnu jsem zvolil, protože je velmi rozšířená a má dobrou dokumentaci. V projektu využívám hudbu 8Bit Music od autora GWriterStudio, načítací kolečka Animated Loading Icons od autora Infima Games, animaci exploze Cartoon explosion BOOM Weapons Gun VFX od autora Eretichable a herní pozadí Pixel Skies DEMO Background pack od autora Digital Moons. Všechny zmíněné prostředky jsou dostupné zdarma v Unity Asset Store. Veškerý zbytek projektu je mou originální prací, včetně všech kódu a vytvořených textur.

5. ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

Mým cílem bylo vytvořit online hru s možností vytváření vlastních předmětů. Online aspekt i vytváření předmětů se mi podařilo implementovat. Jelikož se jednalo o mojí první online hru, musel jsem se seznámit s konceptem programování síťových her. Testování takovýchto her je mnohem složitější než u single-player her a v jedné osobě se hra nedá správně otestovat. Povedlo se mi ale párkrát otestovat hru s přáteli. Hra samotná se sice nemůže porovnávat s obdobnými hrami velkých studií jako je například Brawhalla, obsahuje však unikátní možnosti a to zejména vytváření vlastních předmětů přímo ve hře. Za nejnáročnější část práce považuji implementaci algoritmu pro tvorbu kolizních těles a synchronizaci herních světů všech hráčů v místnosti.

6. INSTALACE

Projekt lze stáhnout ze stránky Github na adrese: https://github.com/JirkaBitt/Maturitni-Projekt.git. Na této stránce lze stáhnout zip soubor obsahující projekt. Po stažení souboru lze v aplikaci Unity Hub otevřít stažený soubor, Unity Hub poté projekt otevře. Projekt obsahuje šest scén, hlavní scéna má název ChooseLevel.

Součástí staženého souboru je i sestavená hra, která lze spustit jako aplikace.

Hotový projekt vyžaduje minimálně 1,5Gb RAM, na mém počítači s procesorem M2 Pro využíval při běhu 33% CPU a 21% GPU.

7. OVLÁDÁNÍ

7.1. Herní menu

Po zapnutí se zobrazí herní menu. Zde se uživatel může rozhodnout, zda vytvoří nové předměty a novou hru, načte uložené předměty, nebo se připojí ke hře jiného hráče. Pokud se uživatel rozhodne připojit se k jinému hráči, potřebuje znát ID herní místnosti, do které se chce připojit. Jakmile toto ID zná, může ho zadat do textového pole a zmáčknou tlačítko pro připojení se do hry.

7.2. Vytváření předmětů

U vytváření předmětů může hráč přepínat mezi tužkou a gumou tlačítky v levém horním rohu. Kreslit se dá levým tlačítkem myši, průměr nástrojů se mění otáčením kolečka myši. Po nakreslení všech předmětů se zobrazí přehled všech předmětů. Zde může uživatel kliknout na vybraný předmět a dodatečně ho upravit. V tomto přehledu se také nachází textové pole, kam uživatel může zadat svoje herní jméno, které se pak ukáže ve hře. Hráč může zahájit hru zmáčknutím tlačítka pod zmíněným textovým polem.

7.3. Hra

Herní postava se ovládá tlačítky A pro pohyb doleva a D pro pohyb doprava. Tlačítkem Space se skáče, Shift slouží pro použití dovednosti sprint. Pokud hráč chce vzít nebo upustit zbraň, udělá tak tlačítkem E, zbraní se pak útočí levým tlačítkem myši. Tyto informace o ovládání herní postavy se dají zobrazit i ve hře zmáčknutím tlačítka v pravém horním rohu. Pokud chce hráč opustit hru, může tak udělat zmáčknutím tlačítka v levém horním rohu.

7.4. Výsledky

Po ukončení hry se ukáží výsledky, zde se může hráč rozhodnout, zda chce hrát znovu. Učiní tak zmáčknutím tlačítka Play Again napravo od výsledků. Pokud chce opustit hru, může zmáčknout tlačítko Leave.

8. SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] BAELDUNG. What Are Serialization and Deserialization in Programming. Online. Dostupné z: https://www.baeldung.com/cs/serialization-deserialization. [citováno 25.3.2024]

[2] VIOTTI, Juan Cruz; KINDERKHEDIA, Mital. A survey of JSON-compatible binary serialization specifications. arXiv preprint arXiv:2201.02089, 2022.

[3] ZAHID, Hatim. Medium. Online. Dostupné z: https://medium.com/@hatim.zahid/serialization-and-deserialization-how-data-travels-in-a-computer-network-13f61dc225c4. [citováno 25.3.2024]

[4] MADHAV, Sanjay. Game Programming Algorithms and Techniques. Vyd. 1. Addison-Wesley, 2013. ISBN 978-0-321-94015-5.