**Univerzita Hradec Králové**

**Fakulta informatiky a managementu**

**Katedra informačních a kvantitativních metod**

# Návrh řešení desktopové MMO hry

Bakalářská práce

Autor: Lukáš Pohl
Studijní obor: Aplikovaná informatika, ai3

Vedoucí práce: Ing. Michal Macinka

Hradec Králové Srpen 2019

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 13.8.2019

Poděkování:

Na tomto místě bych chtěl poděkovat vedoucímu bakalářské práce Ing. Michalu Macinkovi za metodické vedení práce, cenné připomínky a kritiku práce.

Děkuji.

## Anotace

Cílem této práce je popsat mechaniky MMO her a nastínit možnou implementaci vybraných herních mechanik. V první části se bakalářská práce věnuje popisu MMO her, vybraných podžánrů a jejich obvyklých principů. V druhé části se práce zaměřuje na rozdíly mezi herními frameworky a enginy a některými jejich zástupci. Pojednává také o operačním systému Windows a frameworku .NET. Třetí část nastiňuje praktické příklady řešení několika základních vybraných herních mechanik a rozebírá základy problematiky síťové komunikace mezi serverem a klienty. Poslední část se týká vjemové stránky her, tedy vykreslování, shaderů, částic a prostorových zvuků. Práce je obohacena o poznatky a volby z vývoje MMO hry WarEternal.

## Annotation

## Title: Designing a desktop MMO game solution

The aim of this bachelor thesis is to describe the mechanics of MMO games and to propose implementation of chosen game mechanics. The first part of this bachelor thesis focuses on subgenres and common principles of MMO games. The second part deals with differences between game frameworks and engines and some of their representatives. It also describes the operating system Windows and .NET framework. The third part outlines practical examples of solving a few chosen basic game mechanics and focuses on the basics of network communication between server and clients. The last part is dedicated to the perceptual part of games, such as rendering, shaders, particles, and spatial sounds. The bachelor thesis is enriched by findings and choices from the development of MMO game WarEternal.

# Obsah

# Seznam obrázků

# Seznam tabulek

# 1 Úvod

Způsobů, jakými lidé tráví svůj volný čas, je nepřeberné množství a vznikají stále nové formy moderní zábavy. V průběhu technologického vývoje a globálního rozšíření internetu došlo také k masivnímu vzestupu herního průmyslu. Moderní zařízení jsou schopna vykreslovat a simulovat komplexnější a náročnější scény než kdykoli předtím. Zvyšující se rychlost přenosu dat a nízká odezva umožnily rozkvět her pro více hráčů (multiplayer). Ohromnou částí herního průmyslu dnešní doby se tak staly právě MMO hry.

MMO (Massively Multiplayer Online) hra je online hra s velkým množstvím hráčů připojených ke stejnému serveru. Hráči jsou schopní komunikace a vzájemné interakce. Mohou tak vytvářet skupiny a rozsáhlá společenství, spolupracovat nebo bojovat proti sobě. Počet hráčů MMO s každým rokem neustále stoupá a přibývají nové inovativní MMO tituly. Mezi silné zástupce MMO patří MMORPG a MOBA hry.

MMORPG (Massively Multiplayer Online Role Playing Game) jsou online hry na hrdiny s velkým množstvím hráčů. Hráči si vytvoří svoji vlastní postavu, zvolí její zaměření a upraví jí vzhled. Tato postava poté žije v herním světě, zdokonaluje se a hráče sociálně zastupuje. Postavu unikátní pro daného hráče nazýváme avatar. MMORPG hry jsou často bohaté na historii herního světa a množství aktivit, které nabízejí. Nejznámějším příkladem MMORPG je World of Warcraft od společnosti Blizzard Entertainment.

MOBA (Multiplayer Online Battle Arena) jsou online bojové arény pro více hráčů. Zakládají se na vzájemných střetech týmů hráčů. Ti si volí z velkého množství zpřístupnitelných postav, které se na rozdíl od MMORPG časem dále nezdokonalují. Zvolenou postavu ovládají v rámci jednoho zápasu. Masivním MOBA titulem je například League of Legends od společnosti Riot Games.

Stěžejní předností těchto her, zejména MMORPG, je přítomnost tisíců jiných hráčů a navazování sociálních vztahů s nimi. Tyto sociální interakce u spousty lidí vybudují mnoholetá pouta k danému hernímu titulu a mnohdy doživotní pouta k žánru samotnému. Některé MMO jsou plně zaměřeny na simulaci virtuálního světa a sociální aspekty. Známým příkladem může být Second Life. Pro hráče, kteří mají problémy navazovat mezilidské vztahy, přináší takové tituly jednoduše dostupnou formu relaxace nebo úniku od reality. Jiné MMO hry se zaměřují více na dobývání herních žebříčků,

osvojování si území, masivní souboje nebo simulaci ekonomie a obchodu. Náročnost na schopnosti hráčů a komplexita herních mechanik a simulací se mohou drasticky lišit. Bez rozdílu jsou však všechny vystaveny na společných základech mezilidské interakce.

# 2 Cíl práce

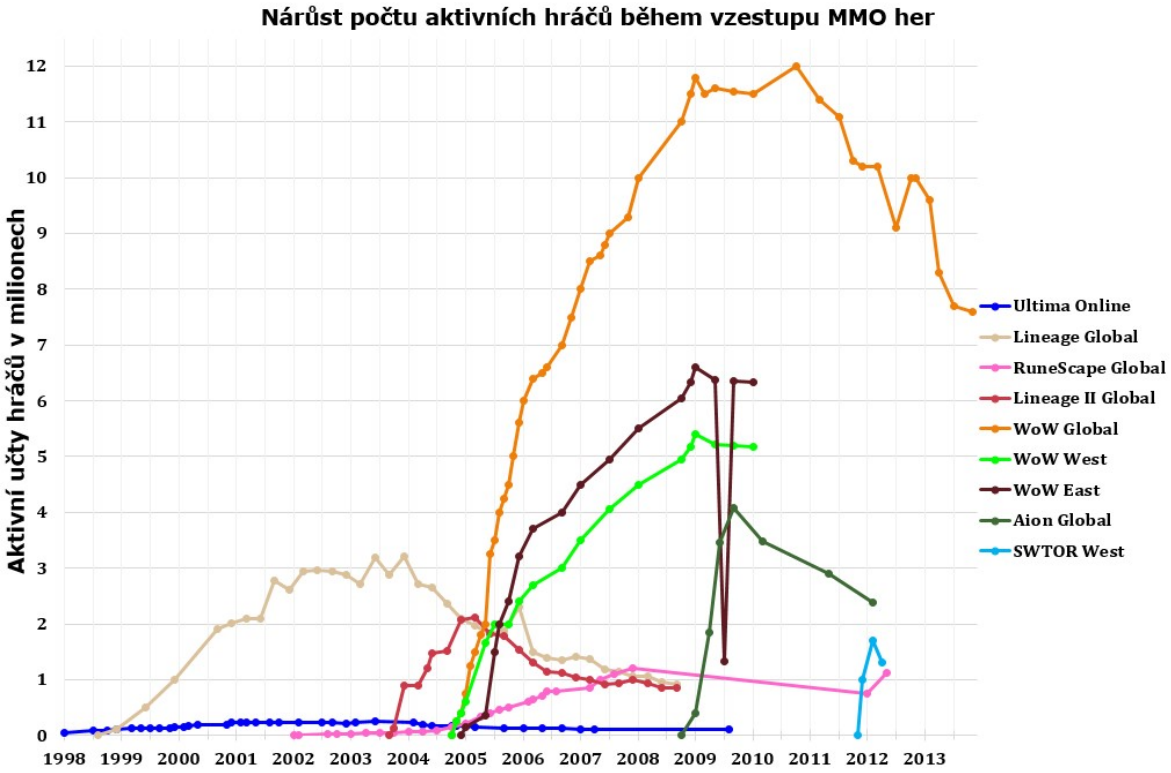
Cílem práce bude popsat problematiku herních mechanik a síťové komunikace u MMO her, nastínit možná řešení vybraných herních mechanik a popsat síťovou komunikaci mezi serverem a klienty. Dále se práce zaměří na vnímanou část her, tedy vykreslování, shadery, částice a prostorové zvuky.

WarEternal je fantasy počítačová hra kombinující elementy z žánrů MMORPG a MOBA. MMO WarEternal je mnou sólově vyvíjeno již pět let a v rámci této práce bude sloužit jako ukázkový projekt. Práce tak bude obohacena o náhledy ze hry, poznatky a volby, které jsem v průběhu jejího vývoje učinil.

MMO hry mohou nabývat nepřeberného množství podob a mohou obsahovat spousty rozmanitých mechanik. Pro zúžení komplexity, množství teorie a názvosloví se práce bude zabývat zejména fantasy žánrem MMORPG a MOBA.

# 3 MMO hry

Od roku 1997 spolu s příchodem hry Ultima Online, započal masivní rozmach MMO her [1]. V roce 2004 vyšel herní titul World of Warcraft, během prvních 24 hodin bylo prodáno přes 240 tisíc kopií a hra brzy dostáhla půl milionu průměrného počtu připojených hráčů [2][3]. Přibývají stále nové MMO hry inovující mechaniky žánru a zdokonalující jejich grafickou stránku i hratelnost. Tento rozmach MMO her ilustruje Obrázek 1.




**Obrázek 1 Nárůst počtu aktivních hráčů během vzestupu MMO her. Zdroj: [4]**

## 3.1 PvE vs PvP

Všechny hry lze rozdělit na PvE a PvP, případně hybrid obojího. V některých herních titulech je PvP odděleno od PvE herního světa formou arén. V jiných jsou herní servery rozděleny na PvE a PvP. Mohou také existovat specifikované oblasti, v nichž je PvP aktivní.

PvE (Player versus Environment, hráč proti prostředí) je typ herního módu, ve kterém neexistují agresivní konflikty mezi hráči. Hráči tak hrají sami, kooperují s ostatními nebo soupeří pouze s počítačem ovládanými nepřáteli. Jedinec slabého a početně se

vyskytujícího druhu takových nepřátel se nazývá mob. Hlavní nepřítel, který je obzvlášť silný a většinou se vyskytuje pouze na jediném místě, je nazýván boss.

PvP (Player versus Player, hráč proti hráči) je herní mód vystavěný na soubojích mezi hráči. Soupeřit mohou jedinci nebo skupiny. V některých hrách se výhrou či prohrou aktualizuje jejich pozice v žebříčku a snaží se tak dobýt co nejvyšších příček. V jiných je jim dovoleno vykrást vybavení nebo materiály poražených nepřátel.

PvE hry mohou mít specifikované oblasti určené pro PvP interakci. V jiných případech jsou servery pro PvP a PvE zcela oddělné a hráči si mohou vybrat. Alternativou může být vyzvání k duelu, kde musí vyzvaná strana s duelem souhlasit. Záškodnické zneužívání PvP k obtěžování nebo zpomalování postupu jiných, často začínajících hráčů se nazývá griefing (odvozeno od grief – nepříjemnost). Přepadení ve větší skupině je označováno jako ganking (odvozeno od gang up, čili spiknout se, spolčit se).

Existují však i různé subkategorie jako například mód WvW (World versus World, svět proti světu) implementovaný hrou Guild Wars 2, ve kterém spolu soupeří velké skupiny hráčů z různých serverů (zde označených jako světy).

## 3.2 MMORPG vs MOBA hry

MMORPG je subžánr MMO her zaměřený ve většině případů více na PvE. Často obsahují také PvP aspekt. Avatary hráčů postupem času získávají lepší vybavení, díky kterému jsou schopni zdolávat náročnější instance. Hráči tvoří společenství, plní úkoly a účastní se herních událostí.

MOBA je subžánr, ve kterém hráči hrají oddělené instance PvP soubojů. Pokud hra umožňuje získávání vybavení a úrovní v rámci souboje, hráč si je neuchovává při přechodu do souboje dalšího. Trvalé výhody získává hráč formou zvyšované úrovně účtu a s ní spojeným odemykáním herního obsahu nebo stromů pasivních bonusů.

## 3.3 Rasy postav

Rasa je prvním ze dvou základních faktorů určujících typ postavy hráče. Různé MMO světy jsou obývány rozmanitými rasami. Třemi nejčastějšími rasami jsou fantasy žánru jsou:

lidé, elfové a trpaslíci. Z jedné rasy může také být odvozeno několik různých ras, jako jsou například lesní, temní, noční nebo krvaví elfové [5].

Zástupci dalších méně obvyklých ras mohou být nemrtví, orkové a hobiti. Spousta her je obydlena i zcela unikátními (nebo alespoň jedinečně pojmenovanými) rasami. Svět Tamrielu, z herní série The Elder Scrolls, obývají například argoniani, rasa humanoidních ještěrů, a khajiiti, rasa humanoidních kočkovitých šelem. Obdobně svět Warhammeru obývají třeba skaveni, rasa humanoidních krys.

Jednotlivé rasy poskytují specifické výhody nebo rasové vlastnosti. Zajímavými příklady jsou dýchání pod vodou, schopnost vidět ve tmě, menší utrpěné poškození při pádu z vysokých míst, imunity vůči elementům a široká spektra bojových vlastností. V některých MMO hrách k rasovým vlastnostem patří dokonce i znalost specifických jazyků, kterým ostatní rasy nemusí rozumět (tedy někteří hráči mohou v chatu vidět zprávy zašifrované) nebo drobné výhody typu rychlejšího sběru přírodních surovin (bylin, získávání dřeva nebo těžby rud).

## 3.4 Třídy postav

Třída označuje z největší části bojové zaměření postav hráčů. Jedná se obvykle o válečníky, mágy, lovce, léčitele, druidy, paladiny, nekromancery, mnichy, barbary a podobně. Každá třída určuje, které (převážně bojové) aktivní a pasivní schopnosti se hráči s rostoucí úrovní zpřístupní [6].

Třída postav mnohdy také stanovuje zdraví a jiné aspekty. Někdy jsou tyto atributy naopak dány jejich rasou.

## 3.5 Úrovně postav

Úroveň je číslo označující míru pokročilosti postavy. Toto číslo je úzce spojeno se sílou postavy, neboť při dosažení každé další úrovně se avatar stává silnější prostřednictvím zvýšených atributů a zpřístupněných nových schopností [7].

Pro dosažení nové úrovně musí hráč obvykle získat určené množství zkušenostních bodů. Počet zkušenostních bodů potřebných pro získání další úrovně mnohdy velice strmě

narůstá. V některých hrách tak hráč dosáhne dvacáté úrovně během prvního dne hraní, zatímco přestup z předposlední na poslední úroveň může zabrat i měsíce.

Zkušenostní body hráč získává různými způsoby. Mezi nejčastější z nich patří eliminace nepřátel, plnění úkolů, procházení instancí, odhalování neprozkoumaných oblastí herního světa, výroba předmětů nebo časově omezené herní události.

V některých hrách se atributy postavy při dosažení nové úrovně samy zvýší a hráči se zpřístupní nové schopnosti. V jiných hrách hráč získá atributové a dovednostní body, které může utratit za naučení nových dovedností. Některé hry dokonce vyžadují, aby hráč pro získání nové úrovně po dosažení potřebného počtu zkušenostních bodů se svým avatarem nejdříve ulehl ke spánku. V jiných je třeba vyhledat specifické NPC (Non Player Character, počítačem ovládaná postava), u kterého se za dovednostní body (případně i dané množství herní měny) lze nové schopnosti naučit.

Existují ale také hry, které naopak zlepšování daných dovedností staví na četnosti jejich používání hráčem. Získání úrovně pak závisí na zlepšení stanoveného počtu dovedností, načež hráč získá několik rozdělitelných atributových bodů.

## 3.6 Atributy postav

Atributy označují vlastnosti postavy a jejich výše je reprezentována nějakou hodnotou. Jedná se o charakteristiky, jež se s rostoucí úrovní postavy zlepšují a umožňují hráči podstupovat výzvy, kterým by dříve nebyl schopen čelit [8].

Mezi atributy, které nejsou většinou přístupné pro přímé zlepšení použitím atributových bodů patří životy a mana. Životy reprezentují zdraví postavy a jejich klesnutí na nulu znamená smrt. Mana znázorňuje míru magické energie, kterou využívá pro sesílání kouzel či využívání speciálních schopností. V některých hrách je přidána také výdrž, která je využívána pro některé schopnosti nebo akrobatické úkony a sprint. Tyto atributy se mohou samy v různých mírách regenerovat a být v nouzi doplněny konzumací lektvarů nebo použitím jiných spotřebovatelných předmětů.

Mezi ovlivnitelné atributy postav patří například síla, obratnost, inteligence, moudrost a vitalita.

Síla obvykle určuje množství poškození, které je hráč schopen udělit zbraněmi na blízko a váhu vybavení, jež je schopen nést. V některých případech může mít síla vliv na množství zdraví nebo míru imunity vůči živlům a jedu.

Obratnost může určovat množství poškození se zbraněmi na dálku, přesnost / šanci na zásah, šanci na kritický zásah (zásah udělující mnohonásobné poškození), schopnost plížit se nedetekován nepřáteli apod.

Inteligence a moudrost mají vliv magické aspekty postavy, tedy množství kouzel, jež si je schopna pamatovat, jejich sílu, zasáhnutou oblast a množství many, které má avatar k dispozici.

Vitalita koresponduje s množstvím zdraví, výdrže a v některých případech také poskytuje postavě drobnou míru imunity.

## 3.7 Schopnosti postav

Schopnost je převážně bojová dovednost, kterou je postava od určité úrovně schopna vykonávat [9]. Schopnosti se dělí na aktivní a pasivní, tedy ty, které hráč musí použít manuálně a ty, jejichž efekt je neustále přítomen.

### Aktivní schopnosti

Mezi aktivní schopnosti zařazujeme rozmanité speciální útoky, kouzla, léčení, nástrahu pastí a tak dále. Některé schopnosti mohou na hráče uvrhnout různé pozitivní nebo negativní stavy.

Aktivní schopnosti často vyžadují určité množství many nebo výdrže, bez které je není možno použít.

### Čas aktivace a čas vychladnutí

Každá aktivní schopnost má svůj specifický čas potřebný k aktivaci a čas po který ji není možné znovu použít.

Čas aktivace označuje dobu, po níž se musí avatar hráče koncentrovat na použití dané schopnosti. Během této doby je postava náchylná speciálním přerušovacím útokům, které naruší koncentraci postavy a schopnost tak nebude vyvolána [10].

Čas vychladnutí započne spolu s časem aktivace. Jedná se o časový úsek, po který nelze použít onu stejnou schopnost od jejího posledního použití.

### Pasivní schopnosti

Pasivní schopnosti poskytují hráči různé typy výhod, které většinou není potřeba manuálně používat. To však neznamená, že nemohou mít vnitřní mechaniky využívající časů vychladnutí.

Některé schopnosti jsou koncipovány pasivně jen z části, tedy tak, aby jejich efekt nebyl pro hráče vždy výhodný a museli je tudíž manuálně ve vhodné okamžiky aktivovat a deaktivovat.

## 3.8 Stavy

Stav označuje kondici, ve které se postava hráče, NPC nebo nepřítel nachází. Trvání stavu může být omezeno dobou trvání nebo podmínkou, za níž se stav zruší [11].

Stavy lze obecně rozdělit na pozitivní a negativní. Stav, který nelze označit za pozitivní ani negativní je označován za neutrální.

Pozitivní stavy, někdy označovány jako požehnání, zahrnují zvýšení atributů, statistik postavy (navýšení odolnosti, rychlosti pohybu, šance na kritický zásah atp.) nebo výhody založené na podmínce jako například doplnění části zdraví za každého poraženého nepřítele.

Negativní stavy, také nazývané prokletí, zahrnují snížení atributů a podmínkou vyvolávané negativní události. Dalším druhem negativního stavu je DoT (Damage over Time), tedy stav, během kterého je hráč opakovaně poškozován po dobu jeho trvání. Mezi DoT stavy patří hoření, krvácení a podobně. Dalšími významnými negativními stavy jsou němota, během níž zasažený hráč není schopen používat kouzla, uzemnění, které zamezuje veškerému pohybu hráče a omráčení, které hráči brání v jakékoli aktivitě.

Většinu častých stavů jako je němota a omráčení lze implementovat pomocí počítadla. Pokud je postava ovlivněna alespoň jedním stavem daného typu, počítadlo není rovno nule a herní mechaniky jsou stavu uzpůsobeny. Uzemnění je také možno implementovat dostatečným snížením rychlosti pohybu postavy.

Následující kód nastiňuje možný základ implementace schopnosti s omračujícím efektem. Omráčení je zde implementováno jako kombinace uzemnění, které omezuje pohyb hráče, a němoty, která mu zabraňuje využívat schopnosti.

```
public class Player
{
    public int silence = 0;
    public int silenceProof = 0;

    public int snare = 0;
    public int snareProof = 0;

    public bool IsSilenced => silence > 0 && silenceProof == 0;
    public bool IsSnared => snare > 0 && snareProof == 0;
    public bool IsStunned => IsSilenced && IsSnared;

    public void Update()
    {
        if (!IsSnared)
        {
            Move();
        }

        if (!IsSilenced)
        {
            CastSkills();
        }
    }

    public void Move()
    {
        //Player movement
    }

    public void CastSkills()
    {
        //Cast skills
    }
}

public abstract class Skill
{
    public Player caster;

    public Skill(Player caster)
    {
        this.caster = caster;
    }

    public abstract void Update(float elapsedTime);
    public abstract void Draw();
}
```

```
public class ExampleStunningSpell : Skill
{
    public Player target;

    public float durationLeft = 5000f;

    public ExampleStunningSpell(Player caster, Player target) : base(caster)
    {
        this.target = target;

        target.silence++;
        target.snare++;
    }

    public override void Update(float elapsedTime)
    {
        durationLeft -= elapsedTime;

        if (durationLeft <= 0)
        {
            target.silence--;
            target.snare--;
        }
    }

    public override void Draw()
    {
        //Draw the skill
    }
}
```

## 3.9 Vybavení

Vybavení hráčů zahrnuje zbraně, zbroj a konzumovatelné předměty.

Zbraně lze rozdělit mnoha způsoby. Hlavní rozdělení je dle jejich dosahu, tedy zbraně na blízko, jako jsou meče a sekery, a zbraně na dálku, například luky a kuše.

Zbroje se často dělí na lehké, střední nebo těžké a poskytují tak různou míru ochrany. Například těžší zbroje, na rozdíl od lehčích, mohou více limitovat pohyblivost, rychlost postavy a snižovat šance na úspěšné plížení.

Kategorie zbraní a zbrojí, které jsou postavě přístupné, záleží na její třídě nebo dokonce ve vzácných případech na její rase. Současně musí avatar dosáhnout určité úrovně nebo stanovené výše atributů. Hráči tudíž mohou získat předměty, které nejsou schopni v blízké době či dokonce nikdy použít. Takové předměty mohou uchovávat ve svém inventáři nebo je prodat je NPC či jinému hráči za herní měnu. K překupování předmětů slouží různé implementace aukčních systémů.

Konzumovatelné předměty zahrnují lektvary a jídlo, které hráči v nouzi využívají pro doplnění zdraví, many, výdrže a získání nebo zrušení určitých stavů.

Spousta her také obsahuje systém výroby předmětů, jež umožňuje z materiálů vyrábět různé lektvary, jídla, zbraně i zbroje.

Většinu výbavy hráči získávají formou kořisti, kterou mohou získat z poražených nepřátel. Nejlepší výbava ve hře (nebo materiály potřebné k její výrobě) se velmi vzácně objevuje při poražení hlavních nepřátel nejobtížnějších instancí ve hře (nájezdů).

U výbavy bývá definována její vzácnost sahající od obyčejných, neobvyklých a magických předmětů až po unikátní, epické a legendární.

### Inventář

Inventář označuje omezený prostor, do kterého si hráč uskladňuje své vybavení a obvykle má podobu mřížky. Předměty pak mohou pokrývat jedno či více políček. Pro vybavení, které postava aktuálně používá, jsou vyhrazeny speciální pole a ve spoustě titulů je implementována i možnost druhého záložního vybavení a možnosti rychlého přepínání používané výbavy [12].

Hráči také mívají k dispozici truhlu, do níž mohou odkládat ještě větší množství předmětů, které v nejbližší budoucnosti neplánují použít.

## 3.10 Úkoly

Úkol je hlavním či vedlejším cílem hráče. Úkoly lze rozdělit na časově omezené a neomezené. Časově omezené úkoly po uplynutí určitého času selžou a pro jejich splnění je musí, je-li to možné, hráč znovu přijmout. Podmnožinou časově omezených úkolů jsou opakovatelné úkoly, které se náhodně generují pro každý den nebo týden.

Cíl úkolu může být různý. Mezi nejčastější cíle úkolů patří interakce s NPC v rámci úkolu, prozkoumat či objevit neznámou lokalitu, doručit předměty, nasbírat a následně doručit suroviny, porazit specifikovanou skupinu nepřátel, bránit území před nepřáteli nebo doprovodit NPC do určitého místa a ochraňovat jej po dobu výpravy před nepřáteli.

Odměnou za splnění úkolů bývá kombinace zkušenostních bodů, herní měny a výbavy. U některých si hráč volí odměnu dle vlastního uvážení z několika nabízených variant.

## 3.11 Skupiny hráčů

Ve většině herních titulů mezi sebou mohou hráči v rámci sociální interakce vytvářet různé skupiny, ve kterých spolu mohou spolupracovat nebo důvěrně komunikovat.

### Party

Party jsou krátkodobé menší skupiny, které jsou například pětičlenné a v nichž jsou hráči schopni společně rychleji a bezpečněji procházet herním světem a dobývat méně náročné instance. Jejich velikost je vždy omezená [13].

Náročné instance pro vysoké úrovně často dovolují i mnohonásobně větší skupiny hráčů. Takové skupiny se nazývají nájezdové party a jejich velikost slouží ke kompenzaci strmě stoupající obtížnosti instancí a hlavních nepřátel.

### Společenství

Obdobně mohou hráči tvořit společenství, ve kterých mohou jednotlivým členům přiřazovat hodnosti, tedy role a k nim náležející oprávnění jako například přijímat nové členy. Společenství mají obvykle také svojí dedikovanou oblast nazývanou sídlo společenství. Do této instance mohou vstupovat pouze členové daného společenství. Některé hry umožní za určitých podmínek vstup i jiným nepříslušejícím hráčům. Členové se mohou podílet na specifických úkolech, PvP soubojích mezi společenstvími, vylepšování sídla, jeho dekoraci a dalších aktivitách. Sídlo také může poskytovat sdílený úložný prostor pro předměty, suroviny a herní měnu [14].

Zdokonalování sídla často členům přináší rozmanité pasivní bonusy a obvykle stojí velké množství surovin a herní měny. Takové vylepšení může trvat určitou dobu, někdy až v řádech dnů, kterou jeho realizace potrvá a během níž nelze vylepšovat v sídle nic jiného. V rámci zlepšování sídla bývá obvykle možno také rozšířit maximální možný počet členů společenství.

Společenství jsou jedním z nejdůležitějších aspektů MMORPG her. Vytváří komunitu hráčů, která se stává důvodem pokračovaného hraní i v případech, kdy už hráč získal nejlepší vybavení, které hra nabízí, prozkoumal celý herní svět a opakovaně pokořil

všechny jeho instance či žebříčky PvP. Hra se pak stává místem, ve kterém hráč tráví čas konverzací a hraním s přáteli, které v ní, často v rámci společenství, poznal.

## 3.12 Instance

Instancemi jsou nazývány ty části herního světa, které jsou odděleny a izolovány pro každého hráče nebo partu [15].

Příkladem instancí jsou kobky a vysokoúrovňové nájezdy. Kobky jsou určeny zejména pro party obvyklých velikostí a jejich pokoření může trvat dle zkušeností skupiny, délky a obtížnosti kobky hodinu nebo i déle. Nájezd je instance určená pro hráče na nejvyšších dosažitelných úrovních. Takovéto instance jsou dlouhé, náročné a plné silných nepřátel. Nájezdové party jsou tvořeny i mnohonásobně většími počty hráčů, a i přesto je jejich zdolání nezměrně obtížnější. Odměnou pro hráče je však vzácná kořist, která se může po poražení nejsilnějších nepřátel s malou šancí objevit [16].

Jako instance se chovají většinou i sídla společenství nebo některé úkoly, tedy oblasti, v nichž tvůrci hry vyžadují oddělenost hráče a jeho party od zbytku herního světa.

## 3.13 Platební modely MMO her

Je vhodné, aby MMO hra byla schopna přinejmenším pokrýt náklady na svojí existenci, tedy provoz serverů. Prvním platebním modelem u her, který však není příliš rozšířený u MMO her, je jednorázové zakoupení hry. Jiné MMO hry staví svůj zisk na měsíčních poplatcích za hraní. Obecně se tomuto modelu říká pay to play, tedy plať za hraní.

Druhým platebním modelem jsou mikrotransakce, které lze dále rozdělit na několik kategorií:

1. plať, abys vyhrál (pay to win),
2. plať, abys vypadal lépe (pay to look better),
3. plať, aby sis zpřístupnil věci rychleji (pay to get stuff faster).

Prvním, hráči neoblíbeným typem mikrotransakcí je „pay to win“ čili „plať, abys vyhrál“. Hry tohoto typu hráčům za reálné peníze nabízí předměty a výhody, jež nelze ve hře nijak jinak získat. Takoví hráči nad neplatícími hráči získávají neférovou výhodu. Tento

platební model je rozšířen zejména u mobilních her, avšak lze jej nalézt i v desktopových hrách.

Druhým mikrotransakčním platebním modelem je „pay to look better“ neboli „plať, abys vypadal lépe“. Takovéto hry umožňují zakoupení různých vzhledů, které modifikují vzhled postavy, její výzbroje a schopností, avšak nijak neovlivňují její statistiky. Jedná se tedy o zcela férový platební model.

Třetí variantou je „pay to get stuff faster“ neboli „plať, aby sis zpřístupnil věci rychleji“. Tento platební model je velice rozšířený ve hrách MOBA subžánru. Hráči mohou odemknutí nových postav dosáhnout opakovaným hraním nebo si je zakoupit. Síla postav je v obou případech stejná. Tato varianta se tedy dá označit za šedou oblast férového platebního modelu.

## 3.14 WarEternal jako hybrid MMORPG a MOBA

Koncept WarEternal leží na pomezí MMORPG a MOBA hry. Zpočátku byla hra koncipována čistě jako inovativní, komplexní MOBA, tedy hra čistě zaměřená na PvP. V současné chvíli je však hybridem MOBA a MMORPG žánru.

### PvP a PvE moduly

V PvP módu hry čerpajícím z žánru MOBA jsou hráči dané instance hry rozděleni do dvou nepřátelských týmů nebo hrají každý sám za sebe (free for all). Každý hráč si zvolí jednu z mnoha postav a vlastní kombinaci jejích unikátních schopností. Vítězí ten tým či jedinec, který zůstává naživu poslední. Veškerá rozhodnutí při tvorbě WarEternal jsou cílena tak, aby v mechanikách PvP modulu hry neexistoval žádný faktor náhody (random factor). Kompetitivní aspekt by tak měl spoléhat výhradně na dovednosti hráčů. Náhodné vizuální efekty jsou zde do jisté míry samozřejmě přípustné. Hráči zde prostřednictvím výher bojují o posty v žebříčku.

V PvE modulu hry založeném na MMORPG žánru hráči kooperují v početnějších skupinách. Většina mechanik z PvP módu je přenesena, avšak cílem hráčů se stává dobývat dlouhé instance plné nastražených pastí a mocných hlavních nepřátel řízených definovanými pravidly. Za úspěšné pokoření nájezdu hráči získávají kořist ve formě artefaktů a relikvií (loot), díky kterým mohou učinit své postavy v budoucích PvE

instancích silnější. Tyto předměty se ale vztahují pouze k PvE módu. PvP jimi tedy není ovlivněno a zachovává si naprostou férovost.

### Herní měny

Hra je vystavěna na třech měnách. První měnu získávají hráči v PvP, druhá je zakoupitelná za reálné peníze a třetí měnu lze získat v PvE části hry a za herní události. První dvě měny slouží k odemykání nových postav a jejich vzhledů. Druhá měna slouží pouze k urychlenému získání herního obsahu, jedná se tak o hru přístupnou zdarma (free to play) v níž lze rychleji zpřístupnit herní obsah (pay to get stuff faster).

Na rozdíl od ostatních titulů však vzhledy postav ve WarEternal nejsou pouze vizuální záležitostí. Každý vzhled s sebou současně nese i danou kombinaci schopností postavy.

Pouze třetí měnou lze získávat mistrovské vzhledy, které umožňují hráčům libovolnou kombinaci schopností. Hra tedy odměňuje zručnost hráčů, neboť nejlepší obsah lze odemknout jen a pouze zdárným hraním a nikoli za peníze.

Mimo to je každý týden v PvP modulu volně zpřístupněn základní vzhled malé části postav, takzvaná rotace. Ta umožňuje hráčům vyzkoušet si nové postavy bez nutnosti jejich odemknutí první či druhou měnou. Tato mechanika je v žánru MOBA obvyklá.

Hráči mohou mimo to také tvořit společenství, které slouží zejména pro lepší organizaci, komunikaci a trénink hráčů. Úkoly jsou v titulu přítomny v podobě časově omezených náhodně generovaných úkolů, událostí a hlavních nebo vedlejších cílů v rámci jednotlivých nájezdů.

# 4 Herní frameworky a enginy

Herní frameworky a enginy slouží k usnadnění vývoje her. Obalují základní funkcionalitu a poskytují předpřipravené funkce nebo dokonce celé moduly, z nichž lze hru sestavit.

## 4.1 Herní framework

Framework označuje platformu určenou k vývoji softwarových aplikací [17]. Herní framework poskytuje vývojáři vrstvu abstrakce při různých úkonech jako je zpracování uživatelského vstupu, síťová komunikace, vykreslování textur, přehrávání zvuků, simulace fyzických jevů, umělé inteligence apod. Framework zpravidla neurčuje formu herního cyklu a nabízí tak větší míru volnosti.

### MonoGame

MonoGame je .NET multiplatformní framework vyvíjený komunitou po vzoru již nepodporovaného Microsoft XNA. Veškerý zdrojový kód MonoGame je open-source.

První verze frameworku byla vydána v roce 2009 a v současnosti je nejaktuálnější stabilní verzí Monogame 3.7.1 [18][19]. Framework v současnosti ovládá například zpracování uživatelských vstupů, práci s grafikou na vysoké i nízké úrovni, práci s kamerami a 3D prostorem, prostorovým 3D ozvučením a další. Mezi aktuálně podporované platformy patří následující [20]:

- Windows,
- Linux,
- MacOS,
- Android,
- iOS,
- PS4,
- PSVita,
- Xbox One,
- Switch.

### LÖVE

LÖVE je multiplatformní herní framework pro jazyk Lua s volně dostupnými zdrojovými kódy. Framework se zaměřuje na tvorbu 2D her a vývojářům poskytuje jednoduše použitelné metody pro vykreslování, animace, práci s kamerou a zvukem.

První verze LÖVE vyšla roku 2008 a nyní již existuje rozsáhlá oficiální dokumentace s širokým spektrem návodů od vykreslování a síťové komunikace až po simulaci fyziky [21]. Platformy podporované frameworkem jsou [22]:

- Windows,
- Linux,
- MacOs,
- Android,
- iOS.

## 4.2 Herní engine

Herní engine je software k vývoji her [23]. Na rozdíl od frameworků enginy nabízí grafické rozhraní (GUI), širokou škálu nástrojů, předpřipravených robustně zpracovaných herních mechanik, simulací a vizuálních efektů, z nichž lze hru sestavit. Nevýhodou herních enginů je často vysoká komplexita poskytnutých nástrojů a ztráta úplné kontroly nad kódem.

### Unity

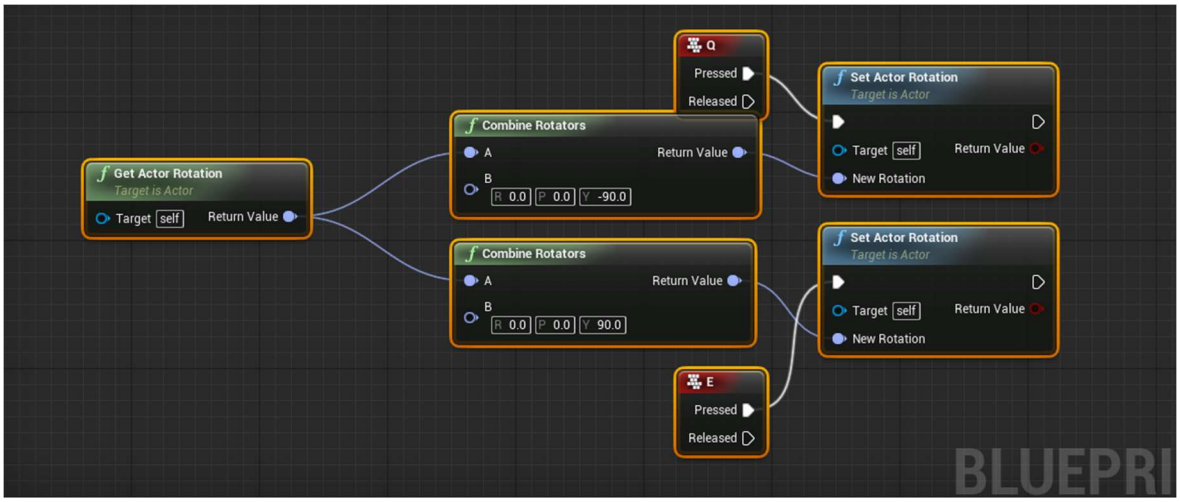
Unity je multiplatformní engine podporující C# a Unity Script, který nese řadu podobností s C++ a Javou. Unity bylo navrženo pro použití vývojáři i grafiky a jeho editor tak umožňuje realizaci množství funkcionality bez psaní vlastního kódu [24]. Hlavní nevýhodou je rozdíl mezi volně dostupnou a placenou verzí. Omezená verze při spuštění hry zobrazuje logo Unity a značně redukuje dostupnou funkcionalitu. První verze enginu byla vydána roku 2005. Současnou stabilní verzí je Unity 2019.1 [25][26]. Mezi současně podporované platformy patří [27]:

- iOS,
- Android,

- Windows,
- MacOS,
- Linux,
- WebGL,
- PS4,
- Xbox One,
- Nintento 3DS,
- Oculus Rift,
- Nintento Switch,
- Vuforia.

## Unreal Engine

Unreal Engine je robustní multiplatformní engine pro jazyk C++. Jeho velikou předností je možnost většinu herní funkcionality sestavit pomocí tzv. modrotisků (blueprints), systému vizuálního skriptování, který vývoj her zjednodušuje, urychluje a současně zpřístupňuje i méně zkušeným vývojářům.




**Obrázek 2 Vizuální skriptování v Unreal Engine pomocí systému Blueprints. Zdroj: [28]**

Unreal Engine zdarma poskytuje veškeré nástroje, zdrojové kódy, podporu platforem a budoucí aktualizace výměnou za 5 % podílu ze zisků. Zástupci podporovaných platforem jsou například [29]:

- Windows,
- Linux,
- MacOS,
- PS4,
- Xbox One,
- HTML5,
- iOS,
- Android,
- Oculus Rift.

## 4.3 Požadavky pro vývoj WarEternal

Frameworky i enginy nabízí rychlejší a pohodlnější způsob, jak rychle prototypovat a vytvářet hry. Kritickou nevýhodou je fakt, že je za vývojáře spousta práce odváděna na pozadí. Z tohoto důvodu byl k vývoji hry WarEternal dočasně zvolen herní framework.

Komplexnost ovládání hry vylučuje mobilní platformy i herní konzole. Od cílové platformy byla také vyžadována maximální rozšířenost. Na základě těchto kritérií je v současné chvíli hra cílena pouze jako desktopová aplikace pro operační systém Windows [30].

WarEternal je vyvíjeno v C# .NET s využitím frameworku MonoGame. Jazyk C# byl k vývoji hry zvolen pro jeho modernost, objektový přístup, jeho použití ve frameworku MonoGame a pro mou mnoholetou zkušenost s ním. Po dosažení funkčního prototypu hry bude od tohoto frameworku pozvolna upuštěno. Hlavním důvodem je odstranění některých nedostatků frameworku, například zamrznutí hry při pohybování s herním oknem, problémy s výkonem nebo neúplná kontrola vývojáře nad veškerým vykonávaným kódem. V současnosti je z frameworku využit pouze základ vykreslování s vlastním kódem pro grafickou kartu a přehrávání zvuků obohacené o vlastní simulaci dynamických prostorových zvuků.

## 4.4 .NET framework

.NET framework je framework pro vývoj a běh aplikací vyvinutý společností Microsoft. Podporuje hned několik programovacích jazyků, které jsou následně překládány do CIL

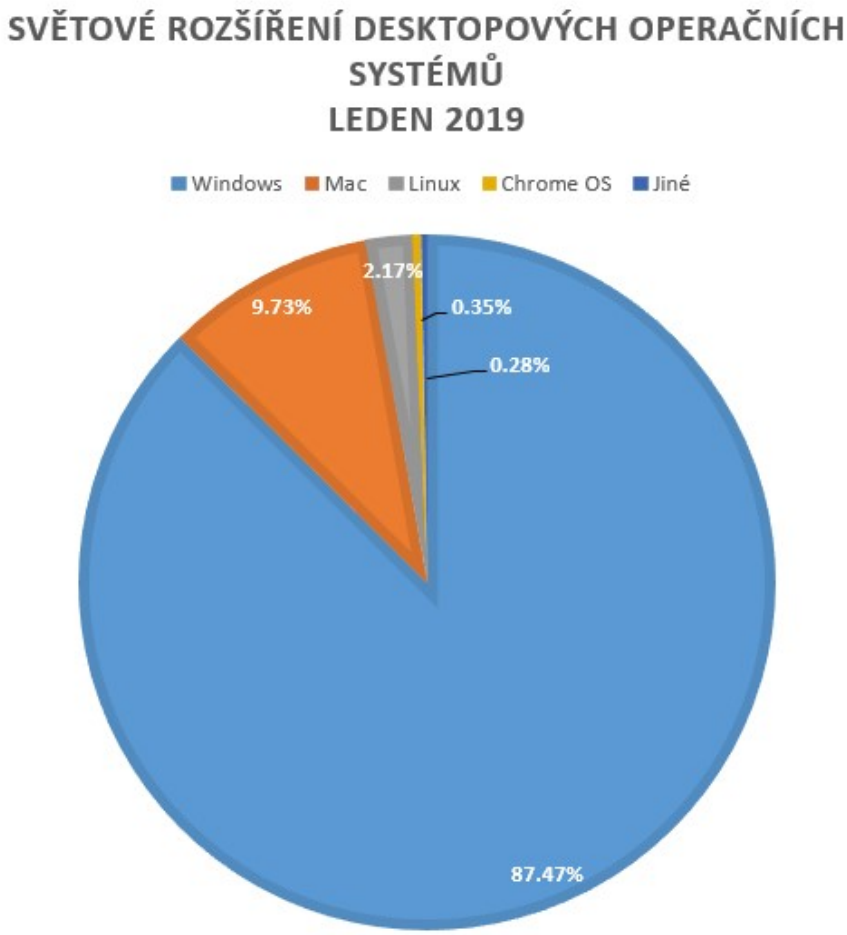
(Common Intermediate Language), hardwarově nezávislého, objektového mezijazyku využívaného .NET frameworkem. CIL je při spuštění programu následně přeložen pomocí JIT (Just In Time) kompilátorem specifické strojové architektury do CLR (Common Language Runtime) pro dané zařízení, z čehož plyne nutnost mít .NET framework nainstalovaný nejen pro samotný vývoj, ale také na cílovém zařízení pro následný běh programu. Mezi rozšířená robustní vývojová prostředí pro .NET patří Microsoft Visual Studio a Jet Brains Rider. Mezi programovací jazyky .NET se řadí například [31]:

- C#,
- C++,
- J#,
- Visual Basic,
- F#,
- JScript.NET.

## 4.5 Microsoft Windows

Microsoft Windows je řada operačních systémů pro desktopová zařízení, ale také tablety a chytré telefony. Upravená verze Windows je také jádrem herních konzolích Xbox One. Windows byl prvním operačním systémem pro IBM (International Business Machiness Corporation), předního výrobce počítačů v US, který poskytoval GUI (Graphical User Interface, grafické uživatelské rozhraní).

Prvním operačním systémem Windows byl Windows 1.0 vydaný v roce 1985. Jednalo se o GUI rozšíření předešlého operačního systému MS-DOS. Pro navigaci bylo možno nově využívat myš a nebylo ji tedy již nutné vykonávat prostřednictvím příkazové řádky. S komerčním úspěchem následovala řada systémů, konkrétně Windows 2.0, 3.0, 3,1, 95, 98, 98 SE (Second Edition, druhé vydání), ME (Millennium Edition, miléniové vydání), XP (Experience), Vista, 7, 8, 8.1 a nejaktuálnější Windows 10 [32]. V současnosti je systém Windows nejrozšířenějším desktopovým operačním systémem [30].




**Obrázek 3 Světová rozšířenost desktopových operačních systémů. Zdroj: [30]**

# 5 Komunikace se serverem

Komunikace mezi jednotlivými klienty her obvykle probíhá prostřednictvím serveru nebo peer-to-peer pomocí TCP nebo UDP protokolu. Různé přístupy mohou mít lepší či horší uplatnění dle typu hry, pro níž jsou využity.

## 5.1 Síť

Síť označuje prostředky potřebné pro navázání spojení, komunikaci a výměnu dat mezi prvky sítě (peer, klient, server). Zařízení v síti jsou identifikována IP adresou a komunikují spolu za pomoci síťových protokolů.

### IP adresa

IP adresa („Internet Protocol address“) je číselná kombinace určená k jednoznačnému rozlišení zařízení v síti. Nesprávným nastavením lze však zapříčinit kolizi adres. Je využívána IP protokolem. IP adresy jsou dále děleny na IPv4 a IPv6.

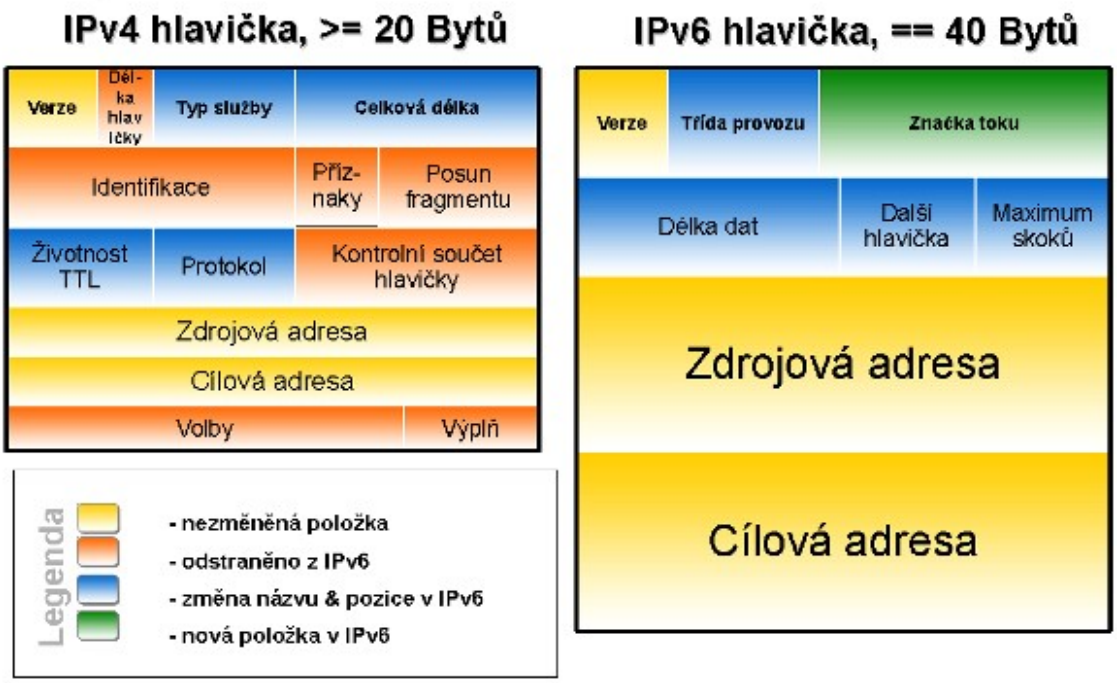
### IPv4

IPv4 adresa je 32bitové číslo zapsané jako čtyři oktety (osm bitů), např. 192.168.0.1. IPv4 adresace je stále využita v drtivé většině sítí. Výhodou IPv4 je větší rozšířenost, lepší podpora a snazší zapamatovatelnost adres.

### IPv6

IPv6 adresa je 128bitové číslo zapsáno jako osm hexadecimálních částí, např. 2001:0DB8:AC10:FE01:0:1:0:1234. IPv6 bylo navrhnuto jako náhrada IPv4 z důvodu nedostatku IPv4 adres.

Výhodami IPv6 je mimo jiné například auto-konfigurace, zamezení kolizí mezi privátními adresami, zjednodušení hlavičky dat, jednodušší a více efektivní směrování paketů, jednoduší administrace a vestavěná autentifikace.




**Obrázek 4 Porovnání hlaviček IPv4 a IPv6. Zdroj: [33]**

### Port

Port je číslo identifikující komunikaci mezi zařízeními. Pomocí portu zařízení rozpozná, které aplikaci jsou data přijatá po síti určena.

Specifická čísla portů jsou rezervována pro určené služby. Prvních 1024 portů je nazýváno „Well-known port numbers“, tedy „Čísla dobře známých portů“.

### Odezva

Odezva (ping) označuje délku trvání komunikace tam a zpět mezi dvěma koncovými zařízeními. Obvykle využívá ICMP (Internet Control Message Protocol) echo request pakety, avšak lze ji nasimulovat a počítat i manuálně.

## 5.2 Síťová architektura

Síťová architektura je design zahrnující hardware, software, protokoly a typ média použitého k přesunu dat. Dvě nejrozšířenější architektury jsou peer-to-peer (P2P) a klient-server.

### Peer-To-Peer

Architektura peer-to-peer (rovný s rovným) neobsahuje hierarchii mezi zařízeními. Všechna zařízení jsou si rovna, komunikují mezi sebou navzájem a sdílejí své zdroje. Peer je současně dodavatelem i spotřebitelem zdrojů [34]. Tato architektura tedy postrádá centralizovanou logiku a je také méně odolná vůči napadení.

Peer-to-peer může tudíž být vhodnou volbou pro hry, jež mají multiplayer mód pouze v LAN (Local Area Network) a nemají globální kompetitivní zaměření.

### Klient-Server

Klient-Server je architektura sítě, která odlišuje klientskou část od serveru. Klientem je často aplikace s GUI, zatímco serverem může být například pouze konzolová aplikace na zařízení s veřejnou IP adresou. Server provádí centralizované výpočty citlivých informací, shromažďuje a distribuuje data a provádí validaci dat přijímaných od klientů.

Klient-Server je vhodnou volbou pro hry s globálním multiplayer módem nebo globálním kompetitivním zaměřením (např. MMO a MOBA hry).

## 5.3 Síťové protokoly

Protokol je standard, podle kterého v síti probíhá přenos dat mezi dvěma koncovými body (dvěma peery nebo klientem a serverem). Protokol mimo jiné definuje např. strukturu dat, zabezpečení a synchronizaci komunikace.

### IP protokol

IP protokol je základní protokol síťové vrstvy poskytující přenos dat. Zařízení rozlišuje pomocí IP adres a sám o sobě nezaručuje úspěšný přenos dat. Stará se o segmentaci paketů do rámců (frame) a následné znovu-sestavování a zajišťuje doručování paketů dle IP adres v hlavičce paketu.

### UDP protokol

UDP protokol, stejně jako IP protokol, nezaručuje spolehlivý přenos dat. Postrádá fázi navázání a ukončení spojení. Důsledkem těchto vlastností je o něco vyšší rychlost oproti TCP protokolu. Využívá čísla portů pro rozlišování jednotlivých aplikací [35].

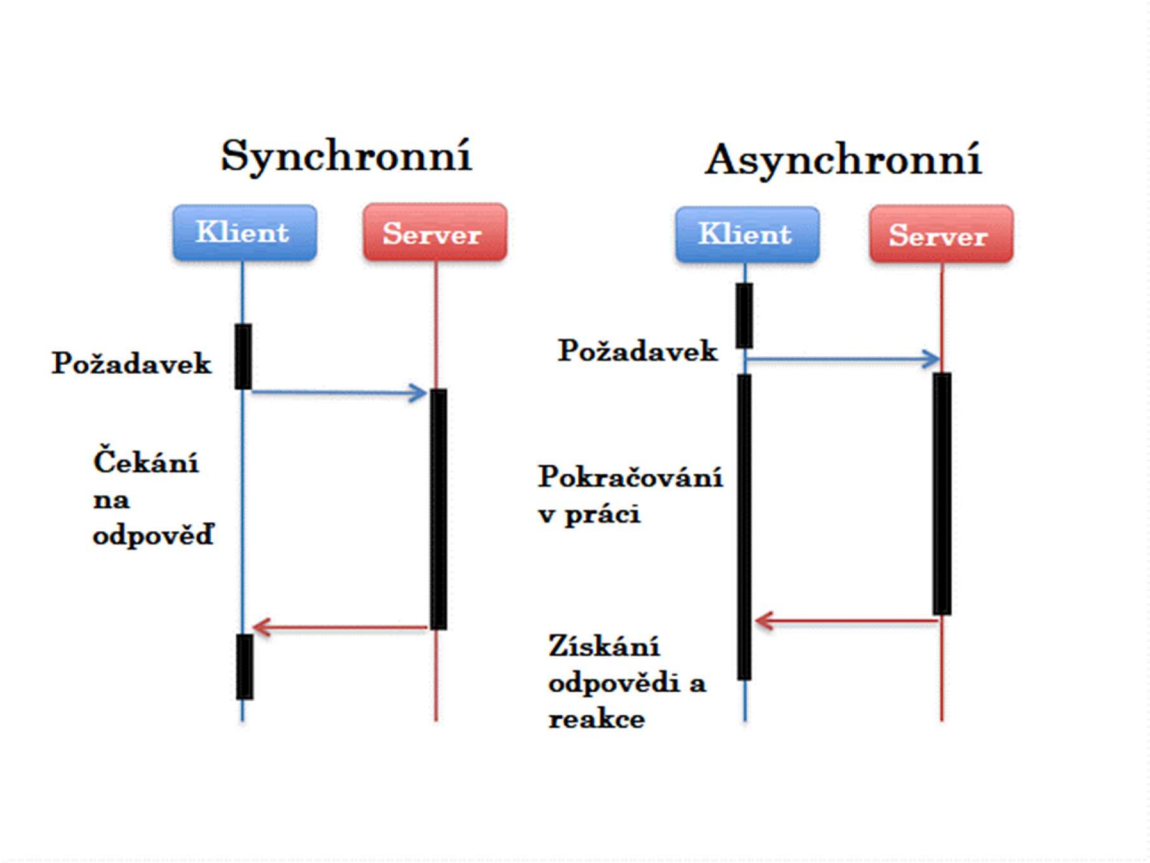
Vhodné využití UDP protokolu je pro případy, kde ztráta dat nezpůsobí žádný problém, například tam, kde by se při opětovném pokusu o odeslání již data staly nevalidními. Ve hrách je proto UDP nejčastěji využito pro pohyb hráčů a jiných entit, neboť se staré pozice časem stávají nevalidní.

### TCP protokol

TCP protokol na rozdíl od UDP navazuje duplexní spojení (současný obousměrný přenos dat). Zaručuje doručení všech dat beze ztrát a ve správném pořadí. Stejně jako UDP využívá čísla portů k identifikaci aplikací. TCP je vhodné použít všude, kde je kritické, aby žádná data nebyla ztracena.

## 5.4 Synchronní a asynchronní komunikace

Komunikace mezi dvěma zařízeními může probíhat synchronně nebo asynchronně. Synchronní komunikace blokuje vlákno. Obvykle proto vyžaduje dedikované vlákno pro každé připojení. Díky této limitaci je špatně škálovatelná pro velké množství připojení. Asynchronní komunikace je neblokující. Touto vlastností se stává vhodnou volbou pro aplikaci využívající většího množství současných spojení. Současně je také znatelně jednodušeji integrovatelná do jakéhokoli scénáře zahrnujícího interakci s uživatelským rozhraním.




**Obrázek 5 Porovnání synchronní a asynchronní komunikace. Zdroj: [36]**

## 5.5 Síťová komunikace ve WarEternal

Volba způsobu komunikace závisí především na typu hry a jejích požadavcích. WarEternal částečně spadá do kategorie MOBA, je to tedy globální kompetitivní hra. Současně vyžaduje centralizovanou logiku pro ověřování dat hráčů jako prevenci proti manipulaci s daty. Vhodnou architekturou je tedy klient-server. Důraz je kladen jak na rychlost komunikace, tak na bezztrátovost dat. Asynchronní komunikace je tedy klíčová.

Nabízí se možnost kombinace UDP pro synchronizaci pohybu a TCP pro veškeré ostatní akce. Pro zpřesnění funkcionality přehrávání záznamů již odehraných her pouze z kolekce paketů je však vhodné použít TCP protokol pro veškerou komunikaci.

Komunikace byla vybudována pro potřeby hry čistě nad .NET Sockety bez použití TcpClient a TcpListener tříd. Toto umožnilo větší množství nízko úrovňové optimalizace.

TCP protokol byl v prvotním testování shledán dostačující s průměrnou odezvou okolo 70-121 ms z České republiky na server umístěný v Amsterdamu. Hráči z USA však zaznamenali mnohem větší odezvu pohybující se mezi 305-1020 ms. Provedená měření ilustruje Tabulka 1. Hodnoty byly měřeny TCP simulací odezvy pro zaručení podmínek co nejvěrnějších herním paketů. Testovací pakety obsahovaly pouze nezbytné množství dat pro rozlišení požadavku a odpovědi a byly posílány rychlostí 20 za minutu v 10 nezávislých testováních. Z tohoto důvodu byl nad komunikací vypnut Nagleův algoritmus.

| X | Odezva (ms) s Nagleovým algoritmem | | | |
|---|---|---|---|---|
| | ČR | | USA | |
| Měření | Min | Max | Min | Max |
| 1 | 71 | 119 | 297 | 990 |
| 2 | 69 | 116 | 295 | 1100 |
| 3 | 70 | 114 | 304 | 1047 |
| 4 | 72 | 117 | 297 | 930 |
| 5 | 71 | 118 | 309 | 1039 |
| 6 | 71 | 117 | 314 | 1021 |
| 7 | 70 | 120 | 308 | 1089 |
| 8 | 72 | 123 | 303 | 1134 |
| 9 | 71 | 124 | 299 | 1010 |
| 10 | 76 | 127 | 304 | 987 |
| 11 | 73 | 121 | 306 | 967 |
| 12 | 68 | 125 | 311 | 980 |
| 13 | 70 | 124 | 306 | 993 |
| 14 | 72 | 122 | 297 | 1003 |
| 15 | 71 | 121 | 301 | 947 |
| 16 | 70 | 119 | 299 | 962 |
| 17 | 70 | 118 | 305 | 971 |
| 18 | 69 | 123 | 307 | 1024 |
| 19 | 67 | 129 | 313 | 1132 |
| 20 | 69 | 124 | 315 | 1073 |

**Tabulka 1 Měřená odezva při použití Nagleova algoritmu. Zdroj: [autor]**

## .NET Socket

Třída Socket poskytuje velké množství metod pro synchronní i asynchronní síťovou komunikaci pomocí široké škály protokolů.

### UdpClient

Tato třída obaluje .NET Sockety a UDP komunikaci v blokujícím synchronním režimu. Jelikož UDP nenavazuje spojení, neexistuje ke klientu třída UdpListener.

### TcpClient a TcpListener

Tyto třídy obalují .NET Sockety a TCP komunikaci. TcpClient umožňuje komunikaci v blokujícím synchronním režimu. Pro navázaní spojení s klientem, je nutné, aby na druhém koncovém zařízení byl přítomen TcpListener (nebo soket) naslouchající nad zvoleným typem protokolu a na správném portu [37].

### Nagleův algoritmus

Tento algoritmus zajišťuje zvýšení kvality TCP/IP sítí redukcí množství paketů, které jsou po síti posílány. Shromažďuje malé pakety, akumuluje je do větších TCP paketů a odesílá je hromadně, čímž méně vytěžuje síť, avšak uměle navyšuje odezvu. Tato aktivita se nazývá Nagling [38]. S UDP protokolem není použitelný.

Nagleův algoritmus má své místo, avšak v aplikaci vyžadující co nejrychlejší bezpečnou komunikaci se v určitých případech může stát spíše přítěží. Po jeho vyřazení měřená odezva pro Českou republiku klesla na 10-15 ms a pro USA na 90-116 ms.

| X | Odezva (ms) bez Nagleova algoritmu | | | |
|---|---|---|---|---|
| | ČR | | USA | |
| Měření | Min | Max | Min | Max |
| 1 | 9 | 16 | 89 | 118 |
| 2 | 9 | 15 | 93 | 117 |
| 3 | 10 | 16 | 91 | 121 |
| 4 | 10 | 15 | 90 | 116 |
| 5 | 10 | 15 | 88 | 114 |
| 6 | 10 | 14 | 87 | 115 |
| 7 | 11 | 15 | 89 | 113 |
| 8 | 10 | 14 | 91 | 115 |
| 9 | 10 | 14 | 92 | 116 |
| 10 | 9 | 15 | 92 | 117 |
| 11 | 10 | 15 | 93 | 115 |
| 12 | 10 | 15 | 90 | 118 |
| 13 | 9 | 16 | 92 | 112 |
| 14 | 9 | 15 | 90 | 116 |

| | | | | |
|---|---|---|---|---|
| 15 | 10 | 14 | 97 | 117 |
| 16 | 10 | 15 | 91 | 119 |
| 17 | 10 | 15 | 89 | 114 |
| 18 | 11 | 15 | 90 | 115 |
| 19 | 12 | 14 | 92 | 116 |
| 20 | 10 | 15 | 87 | 117 |

**Tabulka 2 Měřená odezva bez Nagleova algoritmu. Zdroj: [autor]**

## 5.6 Obecná konstrukce WarEternal paketů

Data všech typů paketů ve WarEternal začínají 16 bity (2 byty) určujícími typ paketu. Tyto 2 byty jsou v kódu mapovány na enum. U paketů odeslaných od klienta na server je přidán timestamp (časová značka) označující přesný čas v dané instanci hry formou 32 bitů (4 byty).

Pakety určené pro pohyb a chat jsou distribuovány serverem všem klientům připojeným do stejné instance hry.

Všechny ostatní typy paketů obdrží navíc 32 bitů, který jednoznačně identifikuje pořadí, v němž se akce z pohledu serveru udála. Díky znalosti pořadí může server i klient v případě obdržení akcí v nesprávném pořadí vyčkat na chybějící informace. Díky přehrávání akcí ve správném pořadí je jednodušší zajistit správnou synchronizaci herního stavu.

Příklad desynchronizace bez číslování akcí: Hráč A je ovlivněn stavem, který zvyšuje veškeré utržené poškození o 50 %. Ve zlomku sekundy obdrží zranění za 20 životů. Hráč A si tedy odebere 30 bodů zdraví. Hráč B, který však informace obdržel v opačném pořadí, hráči A odebere pouze 20 životů a až poté na něj aplikuje stav. Tím vznikne desynchronizace lokálních proměnných.

Samotné očíslovávání akcí však není dostačující, je také zapotřebí, aby veškeré akce, jež hráč vyvolá, vykonal sám u sebe až po obdržení paketu o jejich provedení od serveru, tzn. ve správném pořadí vůči jiným akcím. Tato kombinace přístupů zaručuje správnou synchronizaci proměnných mezi všemi klienty.

### Dynamická data

Dynamická data jsou obvykle řešena 8 extra bity (1 bytem). Tento byte označuje délku dynamických dat, která za ním následují. Pokud mohou dynamická data nabýt délky delší než 255, například v paketu obsahujícím text poslaný v chatu, poté je zapotřebí dalšího bytu (ushort).

## 5.7 Navázání komunikace

Soket na straně serveru je uveden do režimu naslouchání. Na úspěšné připojení klienta je zaregistrováno zpětné volání, tzv. callback, které se pokusí získat připojený soket a vytvořit nového klienta s tímto soketem. Přijatý soket je poté vyčištěn a nad hlavním soketem je znovu vyvoláno naslouchání. Každý klient asynchronně přijímá data nad svým soketem.

### Udržování komunikace

V okamžiku vytvoření nového klienta je mu odeslán první „Hello paket“, který slouží k udržování komunikace naživu a také k manuálnímu výpočtu odezvy mezi klientem a serverem. Klient z paketu obdrží vždy hodnotu poslední odezvy, kterou poté vnitřně využívá pro vyvolávání akcí a vizuální synchronizaci běhu hry.

Hello paket je ze serveru odeslán každou sekundu za předpokladu, že obdržel od klienta odpověď na předešlý Hello paket.

## 5.8 Ukončení komunikace

Pokud je klient ukončen běžným způsobem, obdrží server informaci o zavření soketu, klienta ukončí a odebere. Byla-li však komunikace přerušena násilně, například přerušením procesu klienta či výpadkem sítě, má dvě možnosti, jak tento výpadek zaregistrovat.

Prvním mechanismem je vypršení (timeout) mezi Hello paketem a odpovědí od klienta. Tento časový úsek byl v případě WarEternal stanoven na 12 sekund a po jeho uplynutí je klient odpojen. Druhý mechanismus se nachází v místě, kde server klientům zasílá data. Pokud server v této chvíli zaregistruje, že klient není dostupný, odpojí jej.

## 5.9 Zpracovávání paketů

Síťová komunikace běží asynchronně. Přijímané byty jsou agregovány do pole a jejich celkový počet je uchováván. V každém herním cyklu je pole uzamčeno a jeho obsah projde prvním průchodem, v němž jsou rozpoznány typy paketů z prvních dvou bytů a v případě paketů s dynamickou délkou je vypočtena jejich konkrétní délka. Je-li současné odsazení a délka paketu menší nebo rovna počtu přijatých bytů, paket je považován za úplný. Úplné pakety jsou přesunuty stranou, zatímco neúplné jsou v poli ponechány a posunuty na jeho začátek.

Pole je odemčeno a je do něj znovu možno asynchronně zapisovat přijatá data. V herním cyklu poté nad získanými celými daty proběhne druhý průchod, který zjistí typ paketu a následně skrze jeho enumový ekvivalent vyvolá delegát uložený ve slovníku.

Tyto delegáty jsou do slovníku načteny pomocí reflexe ze správného jmenného prostoru pouze jednou při spuštění klienta / serveru. Starají se o kompletní rozbor přijatých dat svého typu a o veškerou následnou logiku s těmito daty spojenou.

Pokud paket obsahoval identifikátor akce větší než ID poslední zpracované akce + 1, paket akce je uložen do kolekce a vyčkává na přijetí a vykonání všech chybějících akcí. Poté je zpětně vyvolán a z kolekce odebrán.

Následující kód ilustruje zpracovávání přijímaných paketů založené na předcházejícím textu:

```
public class NetworkHandler
{
    public delegate bool HandlePacketDelegate(byte[] data, ref int startingIndex);
    public enum PacketType : ushort
    {
        PlayerMovement,
        SkillCast,
        //...
    }

    public Dictionary<PacketType, HandlePacketDelegate> packetDelegates = new
    Dictionary<PacketType, HandlePacketDelegate>();
    //Initialized with reflection

    public int ProcessReceivedData(byte[] data)
    {
        int dataLength = data.Length;
        int index = 0;
```

```
        while (index <= dataLength - 2)
        {
            PacketType packetType = (PacketType) BitConverter.ToUInt16(data, index);

            index += 2;

            if (!packetDelegates[packetType](data, ref index))
            {
                index -= 2;
                break;
            }
        }

        return data.Length - index;
        //Returns how many bytes belong to incomplete packet
        //These bytes will be moved to beginning of the buffer and newly received
        bytes will be appended to them
    }
}

public interface IPacket
{
    bool Handle(byte[] data, ref int startingIndex);
}

public class PlayerMovementPacket : IPacket
{
    public bool Handle(byte[] data, ref int startingIndex)
    {
        if (data.Length < startingIndex + 14)
        {
            return false;
        }

        ushort playerId = BitConverter.ToUInt16(data, startingIndex);
        float positionX = BitConverter.ToSingle(data, startingIndex + 2);
        float positionY = BitConverter.ToSingle(data, startingIndex + 6);
        float positionTime = BitConverter.ToSingle(data, startingIndex + 10);

        startingIndex += 14;

        Arena.playersById[playerId].HandleReceivedPosition(positionX, positionY,
        positionTime);

        return true;
    }
}
```

## 5.10 Vizuální synchronizace

Vizuální synchronizace na straně klienta probíhá díky znalosti odezvy mezi klientem a serverem. Cílem synchronizace je, aby všichni hráči viděli co nejpodobnější reprezentaci aktuálního dění ve hře.

### Výpočet hodnoty přípustného opoždění

Použitou hodnotou je nejhorší (tj. nejvyšší) odezva z daného množství posledních obdržených hodnot. Nejhorší odezva je poté navýšena o konstantu doby trvání 1 snímku při běhu klienta v 60 FPS (Frames Per Second, snímky za vteřinu), tedy přibližně 16.67 ms.

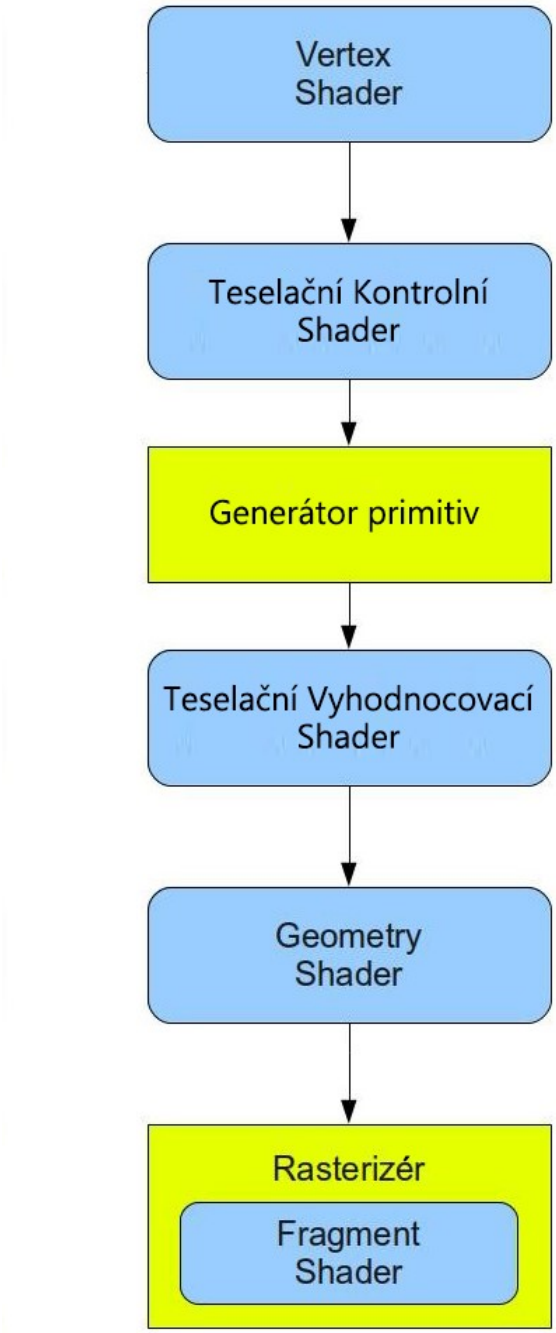
Tato hodnota je dále experimentálně limitována ve vynuceném rozmezí, jež bylo po testování různých hodnot stanoveno do 300 ms. Vyšší hodnoty odezvy přesahují práh podporovaných hodnot, proto by se ideální odezva pro správnou hratelnost měla pohybovat do 300ms.

### Synchronizace akcí a pohybu

Po přijetí paketu s časovou značkou je ke značce přičtena tato dynamicky vypočítávaná hodnota. Pokud se jednalo o akci, je vykonána nebo vložena do fronty, pokud její čas ještě nenastal. Pokud paket poskytoval informace o pohybu, dle výsledné hodnoty je informace o čase a pozici vložena do řazené kolekce. Skrze kolekci je poté v herním cyklu dle času instance hry hodnota pozicí interpolována. Staré záznamy jsou z kolekce postupně odebírány.

# 6 Shadery

Shader označuje program běžící na grafické kartě. Jejich prostřednictvím můžeme přizpůsobovat programovatelné části grafického řetězce. V rámci této práce jsou nejpodstatnějšími základní dva – vertex shader a pixel shader. Posloupnost shaderů ilustruje následující obrázek.




**Obrázek 6 Diagram návaznosti shaderů Zdroj: [39]**

## 6.1 Shader jazyky

Kód shaderů je možno psát v několika různých jazycích označovaných jako shader jazyky. Tyto jazyky jsou uzpůsobeny pro práci s grafickými objekty a obsahují tudíž datové typy jako například vektory, normály, matice a textury, které nám tuto práci značně usnadní. Obsahují také spoustu optimalizovaných matematických operací.

Shader jazyky se dělí na jazyky renderující v reálném čase (Real-time rendering) a jazyky renderující časově náročné scény (Offline rendering).

### Real-time rendering jazyky

Mezi hojně využívané jazyky této skupiny v herním průmyslu patří OpenGL Shading Language (GLSL) a DirectX High Level Shading Language (HLSL) vyvinutý společností Microsoft. Dalším zajímavým jazykem je také Cg programming language od společnosti NVIDIA. Existují také shader jazyky specifické pouze pro určité herní konzole, jako je například PlayStation Shader Language (PSSL).

### Offline rendering jazyky

Vyžadujeme-li komplexní scény v maximální kvalitě, vykreslování v reálném čase nepřipadá v úvahu. Řešením jsou offline renderovací jazyky navržené pro grafiky. Vyžadují tedy nízké nebo dokonce žádné znalosti programování a hardware, neboť jejich cílem není výkon, ale co nejlepší vizuální výsledek.

Vykreslování z důvodu časové náročnosti scén často probíhá ve velkých clusterech počítačů.

Rozšířeným jazykem této skupiny je RenderMan Shading Language (RSL) definující šest hlavních typů shaderů pracujících se světlem, deformacemi, povrchem, objemem (například mlhou) a podobně. Další dva zajímavé jazyky Gelato Shading Language (GSL) a Houdini VEX Shading Language (VEX) jsou blízce založeny na RSL a liší se v mnoha případech pouze syntakticky. Open Shading Language (OSL) vyvinutý společností Sony Pictures Imageworks je dalším robustním zástupcem těchto jazyků a je využíván například v cyklickém rendereru programu Blender.

### Jazyk využitý ve WarEternal

Ukázkový projekt WarEternal je v současnosti zpracován za použití DirectX/HLSL, převážně z důvodu předchozích zkušeností s OpenGL a zájmu rozšířit znalosti a porovnat použití jazyků. Součástí volby je také podpora DirectX ze strany frameworku MonoGame. Oba jazyky si jsou velmi podobné a v budoucnu bude pravděpodobně hra rozšířena o podporu obou.

## 6.2 Vertex shader

Kód vertex shaderu se provádí pro každý vrchol (vertex) vstupní geometrie. Vstupují a vystupují vždy pouze informace o jednom vrcholu. Vstupem kromě vertexu může být například také barva vrcholu nebo uv souřadnice (uv coordinates) reprezentující souřadnice textury, tedy který bod textury bude namapován do daného vrcholu. Vertex shader obstarává například transformace vrcholu v prostoru čili násobení view (pohledovou) a world (modelovou) maticí. Transformacemi uv souřadnic ve vertex shaderu lze simulovat i různé efekty jako například pohybující se vodní hladina.

## 6.3 Teselační shader

Teselace se skládá ze tří částí z nichž pouze dvě, první a poslední, mohou provádět vlastní přizpůsobený shader kód.

### Teselační kontrolní shader

První fází je teselační kontrolní shader (TCS, Tessellation Control Shader), který je ve spojitosti s DirectX nazývaný také Hull Shader. TCS pracuje nad skupinou vertexů, jež v rámci TCS nazýváme kontrolní body (CP, Control Points). Tyto body však namísto polygonální formy definují geometrický povrch definovaný nejčastěji polynomiální rovnicí. Pozice každého bodu tak ovlivňuje celý tento povrch. Lze je v shaderu transformovat, přidávat i odebírat. Skupina kontrolních bodů se nazývá patch. TCS tedy na vstupu přijímá skupinu kontrolních bodů a jeho výstupem je patch.

Shader také vypočítává takzvané teselační úrovně (TL, Tessellation Levels), které určují úroveň detailu (LoD, Level of Detail), tedy kolik trojúhelníků bude patch generovat. Jakým způsobem budou TL počítány je dáno konkrétní implementací shaderu. LoD může záviset

například na vzdálenosti bodů v prostoru nebo na jejich vzdálenosti od pozorovatele, případně kombinaci obou přístupů. Díky tomu může každá patch dle svých vlastností nabývat jiného TL.

### Generátor primitiv

Druhou částí je generátor primitiv (PG, Primitive Generator). Funkcionalita PG je fixní a nelze modifikovat jeho kód. Obstarává generaci geometrických primitivů. Namísto patchí však dle teselačních úrovní rozděluje doménu, která může být definována normalizovaným čtvercem (tj. ve 2D souřadnicích o hodnotách od nuly do jedné) nebo 3D barycentrickými souřadnicemi v případě trojúhelníka.

### Barycentrické souřadnice

Barycentrické souřadnice definují pozici uvnitř trojúhelníku jako kombinaci tří vah směřujících k jeho jednotlivým vrcholům. Součtem všech vah je vždy 1. Například bod umístěný ve vrcholu trojúhelníku bude nabývat hodnoty 1 pro onen vrchol a 0 pro dva zbývající vrcholy. Bod nacházející se ve středu trojúhelníku naopak nabyde hodnoty 1/3 pro váhy všech tří vrcholů.

### Teselační vyhodnocovací shader

Teselační vyhodnocovací shader (TES, Tessellation Evaluation Shader), v DirectX nazývaný Domain Shader, je třetí a poslední částí teselačního procesu. Tato část shaderu má přístup jak k patchím z TCS, tak k souřadnicím z generátoru primitiv. TES je z PG vykonán pro každou souřadnici a jeho úkolem je vygenerovat pro daný bod vertex obsahující pozici a normálu. Tyto vlastnosti lze, podobně jako ve vertex shaderu, transformovat. Každá dokončená skupina tvořící primitiv je pak předána pro rasterizaci.

## 6.4 Geometry shader

Vstupem geometry shaderu je každý transformovaný primitiv (např. trojúhelník). Geometry shader je schopen přidávat nové či ubírat stávající vrcholy (vertexy) nebo jejich pozici měnit. Tímto lze docílit například dynamického LoD modelů v závislosti na vzdálenosti od pozorovatele.

V praxi však geometry shadery využívány moc nejsou, a to díky velmi špatnému výkonu v porovnání s alternativními řešeními. Výjimkou jsou zde grafické karty značky Intel, které se však svojí obecnou výkonností nemohou rovnat kartám NVIDIA a AMD.

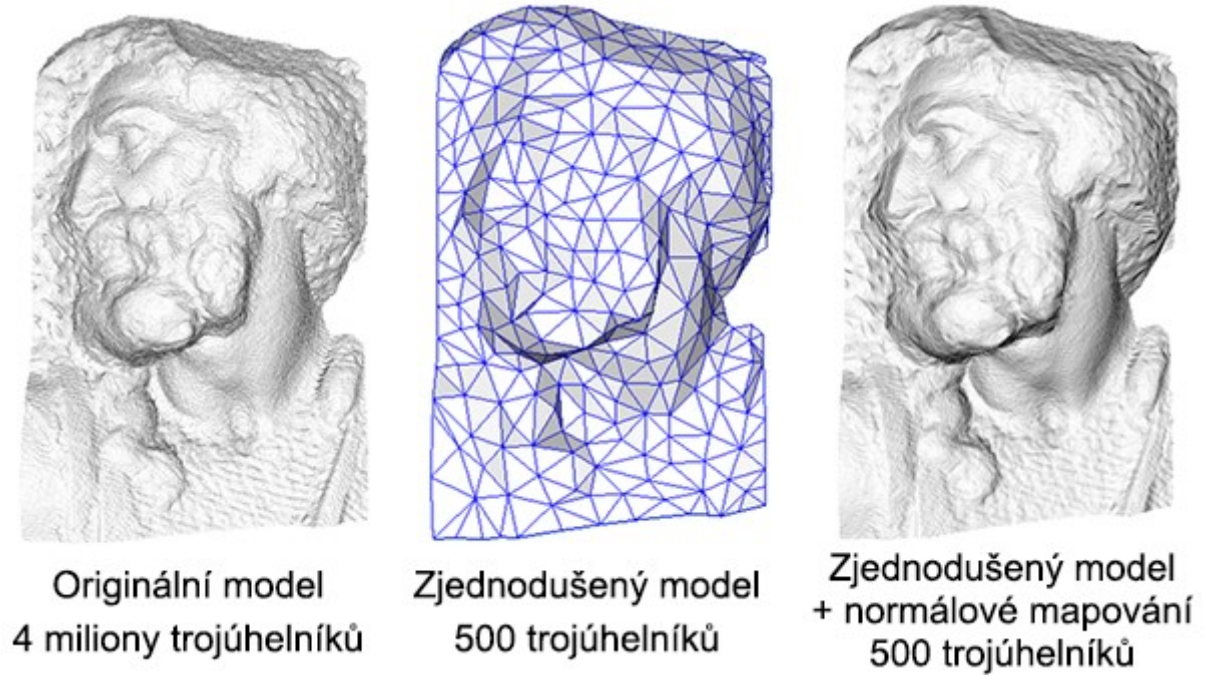
## 6.5 Pixel shader

Kód pixel shaderu (někdy též nazývaného fragment shader) je prováděn pro každý pixel rasterizované scény. Často se tedy zabývá otexturování a modifikací barev daného pixelu. Vstupem pixel shaderu je pozice pixelu na obrazovce (X, Y), uv souřadnice textury, případně barva apod. Barvy a uv souřadnice jsou mezi vrcholy geometrického primitivu interpolovány. V pixel shaderu se také pro daný pixel aplikuje osvětlení, normálové mapování a další vizuální efekty.

### Normálové mapování

Normálové mapování (normal mapping) je technika využívající normálových map. Jako normálovou mapu můžeme použít například texturu, která ve svých barevných složkách obsahuje normalizovaný vektor určující kterým směrem je daný bod na ploše primitiva orientován.

V pixel shaderu lze díky těmto normálám každému bodu modelu přiřadit specifický stupeň osvětlení. Toto vyvolává iluzi mnohonásobně složitějšího modelu při nízké přidané náročnosti na výkon grafické karty.




**Obrázek 7 Porovnání komplexního modelu a zjednodušeného modelu stínovaného pomocí normálových map. Zdroj: [40]**

## 6.6 Použití vlastního shaderu v MonoGame

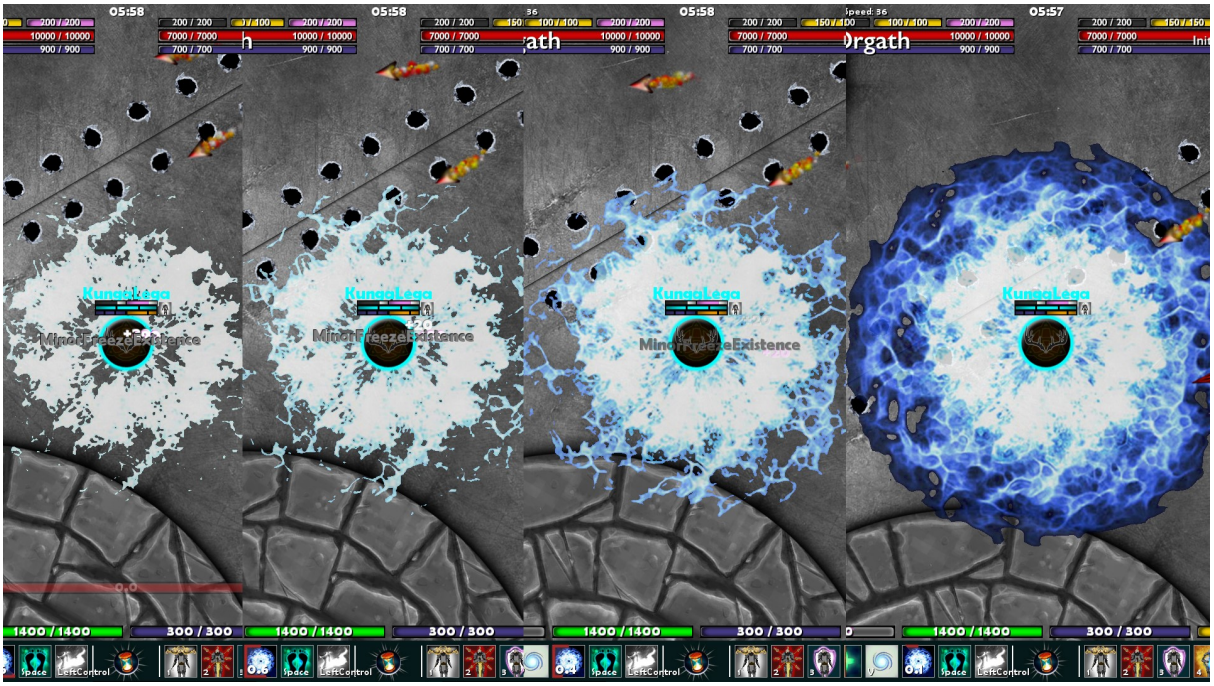
Následující ukázka kódu ilustruje způsob aplikace vlastních shaderů během vykreslení v herním frameworku MonoGame. Třída SpriteBatch ve frameworku MonoGame obstarává shromažďování vykreslovacích volání a následně jejich hromadné provedení. Pokud není potřeba změnit některý z parametrů vykreslování, je vhodné metodu Begin zavolat pouze jednou, provést veškeré vykreslení ve hře a vykreslování dokončit zavoláním metody End.

```
public class ExampleSkillUsingCustomShader
{
    //...
    Effect shader;

    BlendState blendState = BlendState.AlphaBlend;
    SpriteSortMode sortMode = SpriteSortMode.Deferred;

    public void LoadContent()
    {
        shader = content.Load<Effect>(@"Path\RoughSpreadingShader");
    }

    public void Draw()
    {
```

```
        //Start spritebatch – start drawing
        spriteBatch.Begin(sortMode, blendState, null, null, null, null,
        transformMatrix);

        //Set shader parameters
        shader.Parameters["redMultiplier"].SetValue(0.5f);
        shader.Parameters["greenMultiplier"].SetValue(0.5f);
        shader.Parameters["blueMultiplier"].SetValue(0f);

        shader.Parameters["minColorPower"].SetValue(
        KungaMath.TimeInterpolationInverseSin(duration, startingDuration, 1500));
        shader.Parameters["maxColorPower"].SetValue(1f);

        //Apply shader
        shader.CurrentTechnique.Passes[0].Apply();

        //Draw
        spriteBatch.Draw("TextureName", rectangle, origin, rotation, Color.White *
        (0.8f * KungaMath.TimeInterpolationSin(duration, startingDuration, 400)));

        //End sprite batch – finish drawing
        spriteBatch.End();
    }
}
```

Příklad kódu shaderu:

```
sampler mainTexture : register(s0);

float redMultiplier;
float greenMultiplier;
float blueMultiplier;

float minColorPower;
float maxColorPower;

float4 RoughSpreadingShader_P(float4 position : SV_Position, float4 inColor : COLOR0,
float2 mainCoord : TEXCOORD0) : COLOR0
{
    float4 mainColor = tex2D(mainTexture, mainCoord);

    if (mainColor.a == 0)
        return float4(0, 0, 0, 0);

    float colorPower = mainColor.r * redMultiplier + mainColor.g * greenMultiplier +
    mainColor.b * blueMultiplier;

    if (colorPower <= maxColorPower && colorPower >= minColorPower)
        return mainColor * inColor;

    return float4(0, 0, 0, 0);
}

technique RoughSpreadingShader
{
    pass Pass0
    {
        PixelShader = compile ps_4_0 RoughSpreadingShader_P();
    }
}
```



**Obrázek 8 Výsledný efekt ukázkového shaderu měnící se v čase. Zdroj: [autor]**

# 7 Částice

Částice (particle) je objekt zastoupený obvykle malou texturou. Slouží k dynamické vizualizaci efektů jako je například oheň, sníh, déšť, mlha, jiskry, kouř apod. Částice je také možno využít pro simulaci materiálu, jako je tráva nebo srst.

Pro dosažení vyššího výkonu lze využít potenciálu grafické karty za cenu částečné ztráty kontroly nad částicemi. Toto je vhodné využít zejména u těch částic, jichž z jednoho zdroje vystupuje mnoho a které neobsahují žádnou fyzikální interakci se svým okolím. Takovéto částice označujme GPU-heavy čili náročné na grafickou kartu. Ostatní částice jsou tedy CPU-heavy neboli náročné na procesor. Všechny částice podstupují fází simulace a fází vykreslení.

## 7.1 Fáze simulace

Ve fázi simulace je uskutečněn pohyb, akcelerace, rotace, změna velikosti, změna zbarvení, průhlednosti apod. Může zde probíhat také například výpočet kolizí mezi částicí a terénem. Kolize mezi částicemi navzájem nejsou obvykle používány, neboť jsou při velkých množstvích částic výpočetně náročné a nejsou většinou vizuálně znatelné nebo atraktivní.

Ve fázi simulace je také kontrolováno, zda částice nepřesáhla dobu své životnosti. Takové částice jsou poté odstraněny. V případě CPU-heavy částic veškerá simulace chování částic probíhá v logické části herního cyklu.

## 7.2 Fáze vykreslení

Ve fázi vykreslení je vizuálně promítnut aktuální stav částice. Textura je v 3D prostoru vykreslena mezi 4 rohové body tak, aby byla natočena na kameru. Tomuto procesu se říká billboardování (billboarding) [41]. Částice mohou být dále upraveny např. pomocí shaderů. Probíhá ve vykreslovací části herního cyklu.

## 7.3 Zdroj částic

Zdroj částic (particle emitter) má určitou pozici v prostoru a vlastnosti, které předává generovaným částicím. Vlastnostmi, které předává, může být například velikost, směr, rychlost, rotace, textura, barva nebo průhlednost.

Částice generuje v závislosti daného množství za uplynulý čas, přebytek času je poté převeden do dalšího herního cyklu. Toto je obzvlášť důležité u zdrojů částic generujících malá množství částic nebo za předpokladu umožnění běhu hry na různých FPS.

Zdrojem částic ve WarEternal je obvykle schopnost některého z hráčů, hráč samotný (nejčastěji pod vlivem nějakého stavu) či objekt v aréně. Částice jsou také použity ve spouštěči klienta pro generaci kouře a jisker v pozadí.

V nastavení hry je umožněno v reálném čase připínat mezi 4 obecnými množstvími generovaných částic, z nichž první je vyřazení všech nepotřebných částic. Zůstanou tedy pouze takové částice, bez nichž by hráč ztratil informaci o přítomnosti nějaké schopnosti nebo stavu. Konkrétní množství však závisí na každém zdroji částic.


**Obrázek 9 Částice různých schopností ve WarEternal. Zdroj: [autor]**


**Obrázek 10 Částice kouře a jisker ve spouštěči WarEternal. Zdroj: [autor]**

## 7.4 Řazení CPU-heavy částic

Částice mohou být vlastněny zdrojem částic (hráčem / schopností / objektem v aréně) nebo mohou být globální. Vlastněné částice dědí vrstvu pro vykreslení od svého zdroje a jsou vykreslovány napřímo skrze něj.

Globální částice naopak existují nezávisle na existenci zdroje. Mají vlastní vrstvu, podle níž jsou před vykreslovací částí herního cyklu spolu s ostatními vykreslitelnými prvky seřazeny.

## 7.5 GPU-heavy částice

Jedním z přístupů pro GPU-heavy částice je instancování částic (particle instancing). Do vertex shaderu určeného pro částice je předán vertex buffer popisující pouze jediný generický čtverec o délce strany 1 se středem v počátku. Dále je do shaderu předán nově také index buffer nesoucí informace pro každou částici.

Mezi tyto informace patří například celková a zbývající doba života, počáteční pozice, počáteční rotace, počáteční velikost, směr, rychlost, rychlost rotace, změna velikosti

v čase, případně také translace barvy v časy a podobně. Komplexnější translaci barvy může popisovat funkce nebo dedikovaná textura.

Z těchto informací je poté pro každou částici přímo ve vertex shaderu možno vypočítat její správné umístění, rotaci, její velikost, zabarvení apod. Většinou je pro dobrý vizuální dojem žádoucí, aby viditelnost (alfa kanál) a případně také velikost částic spolu s ubývající životností částice dosáhly nulové hodnoty.

Pokud chceme u jednoho takovéhoto zdroje používat více různých textur, můžeme například pixel shaderu předat více textur současně. Ve vertex shaderu poté pro každou částici dle jejího InstanceId (pořadí v index bufferu) můžeme určit například pomocí modula texturu, kterou částice použije. Tuto informaci poté předáme pixel shaderu, který se postará o sampling ze správné textury.

V logické části herního cyklu je pak potřeba v index bufferu udržovat pouze zbývající životnost jednotlivých částic. Mrtvé částice jsou ve vertex shaderu přeskočeny. Seznam takových částic je poté udržován a dle frekvence, s níž zdroj částice emituje, využíván k jejich reinicializaci (dochází k recyklování bufferu). Díky tomu jsou částice značně méně náročné na výkon procesoru. V konkrétní implementaci ve WarEternal je GPU-heavy částic možno vykreslovat přibližně 50x více při dosažení stejných FPS oproti CPU-heavy částicím.

Obstarávání GPU-heavy částic je vhodné obalit do dedikované třídy často nazývané ParticleContainer (čili „kontejner na částice“). Ta si udržuje vertex, index a instance buffer, v logické části herního cyklu obstarává práci s buffery na straně CPU a ve vykreslovací části je předává grafické kartě.

### Řazení GPU-heavy částic

Poloprůhledné částice musí být seřazeny od nejvzdálenější po nejbližší k pozorovateli, aby nedocházelo k vizuálním artefaktům. Tohoto lze ve 2D scéně u GPU-heavy částic docílit jednoduchým vypočítáním Z souřadnice všech vertexů v závislosti na jejich InstanceId. Atribut InstanceId je do shaderu dodán přímo grafickou kartou a není jej tedy nutno manuálně obstarávat. Ve 3D scéně bude brána v potaz reálná vzdálenost částic od pozorovatele, neboť jejich náhodný pohyb v prostoru zaručí různou vzdálenost [42].

Máme také druhou možnost, a to vypnout hloubkové testování. Díky tomu budou částice vykresleny přesně v tom pořadí, v jakém do shaderů dorazí. Toto je vhodné zejména pokud jsme schopni do shaderů dodávat částice už ve správném pořadí bez časově náročného řazení na CPU.

### Hloubkové testování

Hloubkové testování (Depth Testing) je proces prováděný na každém pixelu v pixel shaderu. Vzdálenost vykreslovaného bodu od pozorovatele je převedena do hodnoty mezi 0 a 1 (dle nastavené near a far plane). Tato hodnota je poté testována vůči depth bufferu (Z Buffer).

Depth buffer můžeme reprezentovat jako texturu v níž odstín mezi bílou a černou reprezentuje vzdálenost k nejbližšímu vykreslenému objektu v dané scéně v onom bodě.

Hodnota vypočítaná pro každý pixel je porovnána s hodnotou v depth bufferu. Je-li vzdálenost k pozorovali kratší, bod bude vykreslen a hodnota v depth bufferu nastavena na novou hodnotu nejbližšího bodu.

Brzké hloubkové testování (Early Depth Testing) je hardwarová vlastnost podporovaná většinou dnešních grafických karet. Umožňuje provádět dřívější depth testing ještě přes pixel shaderem. Díky tomu může GPU zcela vyřadit ty trojúhelníky, o nichž ví, že určitě vykresleny nebudou. Vyhne se tak zbytečnému depth testu v pixel shaderu pro každý konkrétní pixel.

### Více systémů GPU-heavy částic

Pro práci s více systémy těchto částic máme dvě možnosti. Lze použít jeden ParticleContainer pro veškeré částice nebo jeden pro každý systém.

### Jediný kontejner pro všechny systémy GPU-heavy částic

Výhodou použití jediného systému částic je, že lze všechny částice dokonale seřadit. Velkou nevýhodou se však stává nutnost udržovat veškeré textury v jediném rozsáhlém atlasu textur.

### Dedikovaný kontejner pro každý systém GPU-heavy částic

Má-li každý systém částic dedikovaný kontejner, částice budou seřazeny pouze v rámci kontejneru. Budou-li se tedy systémy částic překrývat, mohou začít vznikat vizuální artefakty. Variantou pro vyřešení překrývajících se systémů částic může být například kombinace obou těchto přístupů za použití menších, lépe udržovatelných atlasů textur.

# 8 Zvuky

Vhodným formátem zvuků do her je .wav. Jedná se o formát, který nemusí být komprimovaný [43]. Lze od něho tedy očekávat větší velikost souborů. Velkou výhodou nekomprimovaných zvuků je velice krátká doba od načtení k přehrání zvuku. Komprimované formáty, jako například .ogg, které dosahují mnohonásobně menší velikosti, potřebují k dekompresi u dlouhých skladeb až desítky vteřin [44]. Vhodné jsou proto především pro hudbu v pozadí, tedy soubory, které se načítají méně často a v ideálním případě během načítací obrazovky. Zvuky ve hrách tvoří důležitou vjemovou složku. Veškeré instance zvuků lze rozdělit na zvuky statické a prostorové.

## 8.1 Statické zvuky

Statické zvuky nevycházejí z konkrétního místa v prostoru herního světa. Jejich hlasitost i další atributy jsou tak obvykle stanoveny od jejich počátku, v čase jsou neměnné a mnohdy se mezi instancemi stejného zvuku nijak neliší.

Nejčastějším použitím statických zvuků je ozvučení uživatelského rozhraní (například zvuková odezva v okamžiku, kdy uživatel klikne na tlačítko či zvuk tikání odpočtu na začátku zápasu). Takové zvuky jsou monofonické (monophonic). Všechny reproduktory tedy přehrávají totožnou zvukovou stopu.

Druhým typem statického zvuku je hudba hrající v pozadí menu nebo samotné hry. Na rozdíl od statického ozvučení uživatelského rozhraní může být hudba polyfonická (polyphonic), což znamená, že každý reproduktor má vlastní zvukovou stopu.

## 8.2 Prostorové zvuky

Principem prostorového ozvučení je umožnit hráči větší prožitek a ponoření se (immersion) do hry. Současně také prostorové zvuky umožňují hráčům dokonalejší orientaci se v herním prostoru. Skrze manipulaci hlasitosti, vyvážení hlasitosti mezi levým a pravým reproduktorem a případně i úpravy výšky tónů lze vytvořit iluzi zvuku existujícího ve 2D nebo 3D prostoru. Díky tomu je hráč schopen odhadnout kterým směrem se odehrává nějaké dění a kde se přibližně nalézají jeho spoluhráči nebo

nepřátelé. Obdobně lze prostorové zvuky využít také jako vodítka navigující hráče herním světem nebo jako oznámení v blízkosti ukrytých tajemství.

## 8.3 Zdroj zvuků

Zdroj vydávající zvuk (sound emitter) se stává jeho vlastníkem. Může jím být stejně jako u zdrojů částic dynamicky pohybující se hráč, schopnosti či objekt v herním světě. Také jím však může být pouze konkrétní bod v prostoru.

Dle rozdílu pozice hráče vůči pozici zdroje je v logické části každého herním cyklu pro každý prostorový zvuk určena hodnota vyvážení hlasitosti mezi pravým a levým reproduktorem. Obdobně je podle jejich vzdálenosti následně určena také hlasitost zvuku.

Dle typu hry může zvuk v závislosti na vzdálenosti i zcela zaniknout. Některé hry, mezi něž spadá i WarEternal, však pracují s převážně malými herními oblastmi. Hladina hlasitosti jednotlivých zvuků má v takových případech obvykle určenou svoji minimální hranici a vzdálenost od zdroje (radius), za níž jeho hlasitost vždy této minimální hodnoty nabývá. V určité blízkosti je poté stanovena opačně fungující vzdálenost, v níž zvuk vždy nabývá své plné hlasitosti.

Pohyblivé zdroje zvuků pomáhají prostorové zvuky ještě více oživit a vyvolat v hráčích spojení mezi zvukem a jejich zdrojem. Hráči se tak mohou o něco rychleji naučit jednotlivé zvuky rozeznávat a být si alespoň částečně vědomi i těch dění v herním prostoru, které v daný okamžik nevidí, ale pouze slyší.

Následující úryvek kódu je příkladem možného způsobu implementace prostorových zvuků. V každém herním snímku je v metodě Update třídy SoundHandler z rozdílu pozice hráče (posluchače) a zdroje zvuku na základě stanovených konstant pro každý dynamický prostorový zvuk vypočtena nová hlasitost a vyvážení mezi reproduktory.

```
public struct Vector2
{
    public float X;
    public float Y;

    public static float Distance(Vector2 vector1, Vector2 vector2)
    {
        float xDiff = vector1.X - vector2.X, yDiff = vector1.Y - vector2.Y;
        return (float)Math.Sqrt(xDiff * xDiff + yDiff * yDiff);
    }
}

public class SoundEffect
{
    float volume;
    float pan;

    public SoundEffect(float volume, float pan)
    {
        this.volume = volume;
        this.pan = pan;
    }

    public float GetVolume()
    {
        return volume;
    }

    public void SetVolume(float newVolume)
    {
        volume = newVolume;

        //Adjust volume in used sound library
    }

    public float GetPan()
    {
        return pan;
    }

    public void SetPan(float newPan)
    {
        pan = newPan;

        //Adjust pan in used sound library
    }
}
```

```
public class SpacialSoundEffect : SoundEffect
{
    public Func<Vector2> positionFunction;

    public SpacialSoundEffect(Func<Vector2> positionFunction, float volume, float pan)
    : base(volume, pan)
    {
        this.positionFunction = positionFunction;
    }
}

public class SoundEffectHandler
{
    public const float hearingRadius = 5000f;
    public const float maxPanRadius = 1000f;

    public List<SpacialSoundEffect> spacialSoundEffects = new
    List<SpacialSoundEffect>();

    public void Update(Vector2 listenerPosition)
    {
        for (int i = 0; i < spacialSoundEffects.Count; i++)
        {
            Vector2 soundEmitterPosition = spacialSoundEffects[i].positionFunction();

            float newPan = Math.Min(1f, Math.Max(-1f, (soundEmitterPosition.X -
            listenerPosition.X) / maxPanRadius));
            //Assuming -1 and 1 are min/max pan values

            spacialSoundEffects[i].SetPan(newPan);

            float distance = Vector2.Distance(listenerPosition, soundEmitterPosition);

            if (distance >= hearingRadius)
            {
                spacialSoundEffects[i].SetVolume(0);
            }
            else
            {
                float newVolume = (hearingRadius - distance) / hearingRadius;
                //Assuming 0 and 1 are min/max volume values

                spacialSoundEffects[i].SetVolume(newVolume);
            }
        }
    }
}
```

# 9 Závěr

V kapitole „MMO hry“ byly vysvětleny časté mechaniky MMO her a jejich platební modely. Byla zde ukázána možná implementace ukázkové herní schopnosti omračujícího kouzla. Byl zde také přiblížen osobní projekt WarEternal a některé volby při jeho návrhu. Kapitola „Herní frameworky a enginy“ po uvedení do tématu nastínila několik vybraných zástupců obou skupin. Dále se zabývala volbou pro projekt WarEternal a přiblížila využívaný .NET framework a cílový operačním systém Microsoft Windows. Kapitola „Komunikace se serverem“ rozebírala problematiku komunikace mezi klienty a serverem, zpracovávání paketů a vizuální synchronizace mezi hráči. Bylo provedeno měření odezvy v závislosti na použití Nagleova algoritmu a uveden příklad zpracování přijímaných paketů. V kapitole „Shadery“ byly přiblíženy typy a posloupnost shaderů. Bylo zde také ukázáno použití vlastního shaderu v herním frameworku MonoGame. Kapitola „Částice“ se zabývala částicemi, jejich fázemi, zdroji a CPU/GPU variantami. V poslední kapitole „Zvuky“ byly popsány statické a prostorové zvukové efekty, výhody využití prostorových zvuků a jejich zdroje. Kapitola byla doplněna příkladem možného řešení dynamických prostorových zvuků.

Bakalářská práce vycházela z vývoje hry WarEternal. Hra je v aktivním vývoji a do budoucna bude kontinuálně rozšiřována o nové postavy, schopnosti, PvP arény a PvE nájezdy.

Ukázkové video WarEternal: https://bit.ly/2Z4pcYV

Hra WarEternal je mnohaletý osobní projekt a její kód není součástí bakalářské práce.

# 10 Zdroje


[1] Historie Ultima Online [online]. [cit 2019-07-23]. Dostupné z: http://www.uoguide.com/History_of_Ultima_Online

[2] PUTNIK, Goran. Virtual and Networked Organizations, Emergent Technologies and Tools. Vyd. 1. Springer , 2012. 29 s. ISBN 978-3-642-31799-6.

[3] Rekordní prodeje hry World of Warcraft [online]. [cit 2019-04-02]. Dostupné z: https://www.businesswire.com/news/home/20041201005524/en/World-Warcraft-Shatters-Day-One-Sales-Records

[4] Nárůst počtu aktivních hráčů během vzestupu MMO her [online]. [cit 2019-07-30]. Dostupné z: http://users.telenet.be/mmodata/

[5] Rasa postav [online]. [cit 2019-07-30]. Dostupné z: https://wow.gamepedia.com/Race

[6] Třída postavy [online]. [cit 2019-07-30]. Dostupné z: https://wow.gamepedia.com/Class

[7] Úroveň postavy [online]. [cit 2019-07-30]. Dostupné z: https://wow.gamepedia.com/Level

[8] Atributy [online]. [cit 2019-07-30]. Dostupné z: https://wow.gamepedia.com/Attributes

[9] Schopnosti [online]. [cit 2019-07-30]. Dostupné z: https://wiki.guildwars2.com/wiki/Skill

[10] Čas aktivace a čas vychladnutí [online]. [cit 2019-07-30]. Dostupné z: https://wow.gamepedia.com/Cast_time

[11] Stavy [online]. [cit 2019-07-30]. Dostupné z: https://wiki.guildwars2.com/wiki/Effect

[12] Inventář [online]. [cit 2019-07-30]. Dostupné z: https://wiki.guildwars2.com/wiki/Inventory

[13] Party [online]. [cit 2019-07-30]. Dostupné z: https://wiki.guildwars2.com/wiki/Party

[14] Společenství [online]. [cit 2019-07-30]. Dostupné z: https://wiki.guildwars2.com/wiki/Guild

[15] Instance [online]. [cit 2019-07-30]. Dostupné z: https://wiki.guildwars2.com/wiki/Instance

[16] Instance [online]. [cit 2019-07-30]. Dostupné z: https://wow.gamepedia.com/Instance

[17] Definice pojmu framework [online]. [cit 2019-07-23]. Dostupné z: https://techterms.com/definition/framework

[18] První verze herního frameworku MonoGame [online]. [cit 2019-07-23]. Dostupné z: http://www.monogame.net/about/

[19] Aktuální verze herního frameworku MonoGame [online]. [cit 2019-07-23]. Dostupné z: http://www.monogame.net/downloads/

[20] MonoGame [online]. [cit 2019-03-24]. Dostupné z: http://www.monogame.net/

[21] První verze herního frameworku LÖVE [online]. [cit 2019-07-23]. Dostupné z: https://love2d.org/wiki/Version_History

[22] LÖVE podporované platformy [online]. [cit 2019-04-02]. Dostupné z: https://love2d.org/wiki/Game_Distribution#iOS

[23] Definice herního engine [online]. [cit 2019-07-23]. Dostupné z: https://unity3d.com/what-is-a-game-engine

[24] Unity [online]. [cit 2019-03-24]. Dostupné z: https://unity3d.com/programming-in-unity

[25] První verze herního enginu Unity [online]. [cit 2019-07-23]. Dostupné z: https://forum.unity.com/threads/unity-1-0-is-shipping.56/

[26] Aktuální verze herního enginu Unity [online]. [cit 2019-07-23]. Dostupné z: https://unity.com/

[27] Unity podporované platformy [online]. [cit 2019-03-24]. Dostupné z: https://unity3d.com/unity/features/multiplatform

[28] Unreal engine Blueprints [online]. [cit 2019-07-23]. Dostupné z: https://answers.unrealengine.com/questions/107146/simple-movement-blueprint.html

[29] Unreal Engine podporované platformy [online]. [cit 2019-03-24]. Dostupné z: https://unity3d.com/unity/features/multiplatform

[30] Podíl operačních systémů na trhu [online]. [cit 2019-07-23]. Dostupné z: https://netmarketshare.com/operating-system-market-share.aspx

[31] Programovací jazyky .NET [online]. [cit 2019-07-23]. Dostupné z: http://ecomputernotes.com/csharp/dotnet/dotnet-languages

[32] Vývoj operačního systému Windows [online]. [cit 2019-07-23]. Dostupné z: https://www.theguardian.com/technology/2014/oct/02/from-windows-1-to-windows-10-29-years-of-windows-evolution

[33] Porovnání hlaviček IPv4 a IPv6 [online]. [cit 2019-07-30]. Dostupné z: https://www.csirt.cz/page/3099/v-cem-se-ipv6-lisi-a-na-co-bychom-si-pri-jeho-implementaci-meli-dat-pozor/

[34] TANENBAUM, Andrew. Computer Networks. Vyd. 5. Pearson, 2010. 748 s. ISBN 978-0132126953.

[35] KUROSE, Jim a Keith ROSS. Computer Networking - Top Down Approach. Vyd. 6. Pearson, 2012, 199 s. ISBN 978-0132856201.

[36] Porovnání synchronní a asynchronní komunikace [online]. [cit 2019-07-30]. Dostupné z: https://www.smashingmagazine.com/2017/03/simplify-android-networking-volley-http-library/

[37] Třída TcpListener [online]. [cit. 2019-08-11]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistener?view=netframework-4.8

[38] STEVENS, Richard. TCP/IP illustrated, Volume 1: The Protocols. Vyd. 2. Addison-Wesley Professional, 2011, 696 s. ISBN 978-0321336316.

[39] Diagram návaznosti shaderů [online]. [cit 2019-07-30]. Dostupné z: http://ogldev.atspace.co.uk/www/tutorial30/tutorial30.html

[40] Porovnání komplexního modelu a zjednodušeného modelu stínovaného pomocí normálových map [online]. [cit 2019-07-30]. Dostupné z: https://learnopengl.com/Advanced-Lighting/Normal-Mapping

[41] Billboard částice [online]. [cit 2019-07-24]. Dostupné z: http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/billboards/

[42] Instancování částic [online]. [cit 2019-02-04]. Dostupné z: http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/

[43] Formát zvuku .wav [online]. [cit 2019-07-24]. Dostupné z: http://soundfile.sapp.org/doc/WaveFormat/

[44] Formát zvuku .ogg [online]. [cit 2019-07-24]. Dostupné z: https://xiph.org/ogg/

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

**Podklad pro zadání BAKALÁŘSKÉ práce studenta**

| PŘEDKLÁDÁ: | ADRESA | OSOBNÍ ČÍSLO |
|---|---|---|
| Pohl Lukáš | Široká 183/7, Hradec Králové - Věkoše | I1500416 |

**TÉMA ČESKY:**

Návrh řešení desktopové MMO hry

**TÉMA ANGLICKY:**

Designing a desktop MMO game solution

**VEDOUCÍ PRÁCE:**

Ing. Michal Macinka - KIKM

**ZÁSADY PRO VYPRACOVÁNÍ:**

Cíl práce: Popsat problematiku herních mechanik a síťové komunikace u MMO her a navrhnout řešení vybraných herních mechanik.

Osnova:
1. Úvod
2. Cíl práce
3. MMO hry
4. Herní frameworky a enginy
5. Komunikace se serverem
6. Shadery
7. Částice
8. Zvuky
9. Závěr

**SEZNAM DOPORUČENÉ LITERATURY:**

Doplní student.

**Podpis studenta:** Pohl **Datum:** 24.6.19

**Podpis vedoucího práce:** Macinka **Datum:** 24.6.19