FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Online kompetitivní hra

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika

Autor práce: **Vojtěch Linhart**
Vedoucí práce: Ing. Jiří Jeníček, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Online kompetitivní hra

Jméno a příjmení:	**Vojtěch Linhart**
Osobní číslo:	M19000026
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav informačních technologií a elektroniky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Proveďte rešerši herních frameworků a technologií související s online herní komunikací.
2. Vyberte vhodnou technologii a navrhněte modulární architekturu pro web online hru využívají model server-klient.
3. Vytvořte demonstrativní kompetitivní online hru s hráči a počítačem řízenými oponenty.
4. Kriticky zhodnoťte efektivitu a optimalizační potenciál online hry.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] Pine, D.: Learning Blazor: Build Single-Page Apps with WebAssembly and C#, O'Reilly Media, 2022, ISBN 978-1098113247

[2] Zubek, R.: Elements of Game Design, The MIT Press, 2020, ISBN 978-0262043915

Vedoucí práce:	Ing. Jiří Jeníček, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	24. října 2022
Předpokládaný termín odevzdání:	22. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 24. října 2022

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

22. května 2023 Vojtěch Linhart

Poděkování

Tímto bych chtěl poděkovat panu Ing. Jiřímu Jeníčkovi, Ph.D. za vytrvalost při vedení bakalářské práce, za cenné rady a podněty. Také bych chtěl poděkovat své rodině, která mě po celou dobu studia podporovala.

Abstrakt

Bakalářská práce se zabývá problematikou vývoje online počítačových her. Cílem této práce bylo vytvořit demonstrativní online hru pro více hráčů s prvky počítačových oponentů. V práci je popsán vývoj této hry a zhodnocen optimalizační potenciál.

Bakalářská práce nejprve popisuje obecný kontext videoher, stručnou historií videoherního průmyslu, dopad her na jedince a současné trendy ve hraní her. V druhé části jsou uvedeny algoritmy spojené s vývojem online hry a návrh jejich implementace. Práce dále popisuje tvorbu demonstrativní hry, porovnává algoritmy uvedené v teoretické části s reálně použitými a argumentuje jejich užití. Nakonec jsou zmíněny problémy, ke kterým došlo při samotném programování demonstrativní hry a navrženo jejich řešení.

Klíčová slova

Blazor, C#, online hra, server – klient

Abstract

The bachelor thesis describes the problematics of online game development. The goal of this thesis was to create a demonstrative multiplayer online game with computer-controlled opponents. This thesis discusses the development of the game and evaluates its optimalisation potential.

First, there is described a general context of videogames, brief history of videogame industry, impact of playing games on individuals and current trends in gaming. In the second part, there are listed algorithms related with development of an online game with proposition of their implementation. Next, thesis describes the process of development of the demonstrative game, it compares algorithms stated in the theoretical part with algorithms, which are actually used in the game. At last, there are mentioned problems, which occurred during the programming of the demonstrative game, along with suggestion of their solution.

Keywords

Blazor, C#, online game, server – client

Obsah

Seznam obrázků

Seznam zkratek

- DoS – Denial of Service (útok založený na úmyslném přehlcení sítě)
- FPS – First Person Shooter (střílecí hra z pohledu první osoby)
- JSON – JavaScript Object Notation (datový formát)
- HTML – Hypertext Markup Language (jazyk pro tvorbu webových stránek)
- MMORPG – Massively Multiplayer Online Role-playing Game (online RPG pro velký počet hráčů)
- RMT – Real Money Trading (obchodování ve hrách s reálnou měnou)
- RPG – Role-playing Game (hra na hrdiny)
- RTS – Real-time strategy (strategická hra v reálném čase)
- TBS – Turn-based strategy (tahová strategická hra)
- TPS – Third Person Shooter (střílecí hra z pohledu třetí osoby)
- UDP – User Datagram Protocol (rychlý internetový protokol)
- TCP – Transmission Control Protocol (bezpečný internetový protokol)

1 Úvod

V posledních pár desetiletích se videohry staly oblíbenou náplní volného času mnoha lidí. Málokdy si však hráči uvědomí, kolik úskalí a problémů musí vývojář překonat, aby naprogramoval úspěšnou počítačovou či mobilní hru. První hry byly určeny pro jedno zařízení, nicméně s příchodem internetu a zdokonalením online komunikace se vývojářům naskytl nový potenciál – možnost vytvořit hru, kterou může současně hrát více hráčů na oddělených zařízeních. S tím samozřejmě přichází nové výzvy, které musí programátoři překonat – od plynulého přenosu dat až po zabránění podvádění.

Tato práce se věnuje problematice online her a nahlíží na ni z více úhlů pohledu. V první části je popsán obecný kontext videoher – historie, dnešní trendy, žánry videoher a dopad hraní počítačových her na jedince. Druhá část pojednává o problémech spojených s programováním online her. Popisuje algoritmy a porovnává jejich vhodnost použití pro konkrétní situace. Vysvětluje také, s čím se běžný programátor online hry pravděpodobně setká při vývoji své hry, a navrhuje různá řešení. Jsou zde probrány algoritmy online komunikace, představena problematika podvádění v kontextu online her. Dále jsou zde uvedeny technologie pro současnou tvorbu videoher – frameworky a herní enginy.

Ve třetí části je představena demonstrativní online hra, jejíž tvorba byla jedním z cílů této práce. Jedná se o akční online hru s jednoduchou grafikou a nepříliš složitými herními pravidly. Hra primárně slouží k představení a demonstrování popsaných algoritmů. Čtenáři jsou zde mimo jiné předloženy návrhy a rady, pokud by on sám chtěl vyzkoušet naprogramovat vlastní online hru. Tato hra byla vytvořena ve frameworku Blazor, který slouží pro vývoj webových aplikací pomocí jazyka C#.

Na závěr je uveden odkaz na kód demonstrativní hry, kde čtenář může pohlédnout na konkrétní implementaci popsaných algoritmů.

2 Obecný kontext videoher

2.1 Historie videoher

První známky videoher se datují k polovině 20. století. Panuje jistá nejednoznačnost v určení první videohry z toho důvodu, že není přesně definovaný pojem videohra. Pokud se pod slovem videohra chápe hra pro jednoho či více hráčů zobrazována pomocí elektrického zařízení, lze jako první počítačovou hru uvést Cathode-Ray Tube Amusement Device. Tato hra vznikla v roce 1942. Autoři této hry jsou Thomas T. Goldsmith Jr. and Estle Ray Mann. Hra byla inspirována radarem z druhé světové války. Cílem hráče bylo zasáhnout letadlo vystřeleným projektilem. Po úspěšném zásahu letadlo vybouchlo. Jako zobrazovací zařízení byla využita katodová trubice spojená s osciloskopem. Ve výsledku se jednalo o čistě mechanické zařízení bez jakéhokoliv běžícího programu či paměti. Veškerý „program" hry tvořily čistě jen fyzikální děje. Vedou se tedy spory, zdali tento projekt lze považovat za video hru (David S. Cohen 2019a).

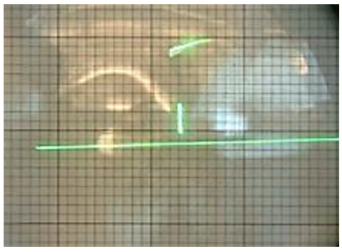
Jako dalšího kandidáta na první videohru lze uvést hru s názvem Noughts and Crosses (přeloženo jako „křížky a kolečka") z roku 1952. Jednalo se o počítačovou verzi hry piškvorek hranou v mřížce 3 x 3 pole. Na rozdíl od předchozí zmíněné hry, zde se již jednalo o počítačový program. Autor hry Alexander Sandy Douglas hru naprogramoval pro demonstraci umělé inteligence. Při hře lidský hráč ovládal křížky a počítač kolečka. Tahy počítače nebyly náhodné, ale reagovaly na tahy člověka.

Hra se nedočkala popularity hlavně z toho důvodu, že mohla být spuštěna pouze na jednom konkrétním počítači EDSAC, který se vyskytoval na univerzitě v Cambridge. (David S. Cohen 2019).

První hra, která se dočkala veřejného úspěchu, byla Tennis for two z roku 1958. Tuto hru vytvořil Americký fyzik William Higinbotham, který spolupracoval na tvorbě první nukleární bomby. Při jedné vědecké výstavě pro veřejnost Williama napadlo, že by bylo vhodné přidat nějakou atrakci pro veřejnost, která by přilákala pozornost. Pomocí jednoduchých elektrických komponent vytvořil hru pro dva hráče s názvem Tennis for two

(tenis pro dva). Hra simulovala tenis – každý z hráčů ovládal jednu stranu hřiště. Pokud se míček nacházel na jejich straně hřiště, mohli ho odpálit a snažit se dostat míček na druhou stranu hřiště při nejlepším tak, aby ho druhý hráč nebyl schopný odehrát zpět. Pravidla hry byla stejná jako při reálném tenisu – jakmile míček skončil u jednoho hráče, hráč prohrál.

Podle výpovědí přítomných se hra stala senzací celé výstavy a zaujala převážně vysokoškolské studenty. (John J. Anderson 1983)



Obrázek 1: Tennis for two

Na obrázku lze spatřit simulaci tenisového kurtu s letícím míčkem
Zdroj: (Brookhaven National Laboratory 2013)

Potenciálu videoher si všimli další společnosti a vývojáři. V roce 1972 se na trh dostala první videoherní konzole. 1977 vzniká masivně populární hra Pac-Man od Japonského vývojáře Tomu Iwatani. Arkádové herní automaty se objevují ve veřejných prostorách.

Kolem 80. let je zaznamenán nárůst vlastnictví osobních počítačů, čeho využívají herní společnosti. Objevuje se značka Nintendo se svými herními konzolemi. Vzniká další ikonická hra Super Mario Brothers.

V 90. letech přichází zdokonalování grafiky, vznikají 3D hry. Na trh se dostávají CD-ROM disky s větší kapacitou paměti než dosavadní zařízení, což skvěle nahrává rozšíření videoher. V tomto období lze zaznamenat první hry pro více hráčů umožňující online komunikaci jako například hry Doom nebo Unreal Tournament. Od té doby nastal kontinuální vývoj videoherního průmyslu. Zdokonaloval se hardware, zlepšovaly se algoritmy pro online komunikaci. Paměti s větší kapacitou, rychlejší procesory a silnější grafické karty přinášely stále nové příležitosti pro vývojáře. Vývoj takto šel neustále dál až do podoby, se kterou se lze setkat dnes. (Ivory 2015)

Další zlomový bod v historii video her znamenalo rozšíření mobilních zařízení, které přinesly zas novou příležitost pro autory videoher (za zmínku stojí známá hra Snake).

Dotykový displej telefonů znovu změnil směr vývoje videoher – již není potřeba ovládat hru pouze pomocí tlačítek, ale od nynějška může být ovládaná celým displejem.

Nakonec je vhodné zmínit i relativně novou technologii – virtuální realitu, která nabízí hráčům zas nový zážitek ze hry.

2.2 Druhy videoher

Od vzniku prvních počítačových her uběhlo více než 50 let a za tu dobu vznikly statisíce videoher. Samozřejmě je přirozené, že se jednotlivé hry od sebe liší. S postupem času vznikaly tedy i oddělené kategorie, do kterých se jednotlivé hry dají zařadit. Tímto způsobem mohou hráči nalézt novou hru z jejich oblíbeného žánru. Nic však neváže vývojáře her, aby se striktně drželi těchto kategorií. Proto je běžné nalézt logické hádanky v akčních hrách či hororové prvky ve strategických hrách. Samotná fráze „herní žánr“ má více významů. Může se tím zamýšlet herní prostředí (historické, sci-fi, hororové…), na jaké schopnosti hráče je hra cílena (logické, strategické, akční…), podle počtu hráčů (pro jednoho hráče, pro více hráčů na jednom zařízení či online…) či podle herního systému (plošinové, bludišťové, otevřený svět, metroidvania…). Nejedná se tedy o jedno dělení, ale o více kategorií, do kterých hra může spadat. Například hra Trine může být označena jako akční plošinová logická fantasy hra.

V následujících podkapitolách budou zmíněny nejopakovanější žánry, podle kterých lze hry rozdělit. Vzhledem k počtu kategorií zde nejsou uvedeny všechny, ale jen ty, které autor považuje za nejpodstatnější.

Akční

Do akční kategorie spadá mnoho různých her od bojových až po závodní či sportovní. Avšak všechny mají jedno společné. Mezi hlavní schopnosti hráče, na které akční hry cílí, patří rychlé a přesné reakce, rozhodovací schopnosti a dobrá koordinace ruka-oko. Nejčastějším prvkem akčních her je ovládání postavy, jejíž cíl je překonat jakési překážky či nepřátele pomocí pohotových akcí. Příklad známé akční hry může být Pac-Man.

Strategické

Strategické hry cílí na rozhodovací schopnosti hráče. Zde se většinou nejedná o rychlé reakce jako spíše o to zvolit správný plán, jak dosáhnout cíle hry neboli zvolit si správnou strategii. Se strategickými hrami se lze setkat i mimo videoherní průmysl. Mohou se zde zařadit i deskové či jiné společenské hry, jako například šachy.

Strategické hry nejčastěji obsahují prvek spravování omezených zdrojů (resource management), kdy se hráči snaží získat určité suroviny pro posun ve hře, jako je stavba budov, verbování vojáků apod. Interakce mezi hráči je nejčastěji prováděna v podobě bitev, kdy hráči kontrolují své jednotky, snaží se ubránit své území a zároveň dobýt cizí.

Počítačové strategické hry lze dále dělit na tahové (TBS z anglického Turn-based Strategy) nebo hrané v reálném čase (RTS z anglického Real-time Strategy). U tahových strategických her se hráč nejprve rozhodne na základě současného herního stavu. Hra nepokračuje, dokud daný hráč neodehraje tah. Poté, co učiní rozhodnutí, je na řadě oponent (ať už lidský či počítačový) a takto se hráči střídají po tazích. Příkladem mohou být již zmíněné šachy nebo hra Civilization VI.

RTS hry se liší v tom, že během hry se nečeká na tahy hráčů, ale hraje se neustále. Všichni hráči soupeří v reálném čase. Zde je možné spatřit lehké provázání s akčními hrami, protože na scénu přichází i bystré reakce hráčů. Mezi RTS lze zařadit například Northgard.

Logické

Hry tohoto žánru cílí na logické uvažování hráče. Hru tvoří množství hádanek a úkolů, které hráči musí pomocí logického uvažování vyřešit. Prvky logických her lze zaznamenat velmi často i u her jiných typů. Například akční logické hry tvoří zajímavou kombinaci, kdy hráč je často vystaven stresující situaci, ve které však musí řešit logické úlohy pod časovým nátlakem. Typickým příkladem logické akční hry může být Portal II.

Logické hry mimo jiné tvoří velké procento her pro mobilní zařízení. Často se jedná o hry s relativně jednoduchými logickými úkoly, u kterých se člověk snadno odreaguje, aniž by se příliš vyčerpal nalézáním správného řešení.

RPG, hra na hrdiny

Název tohoto žánru vychází z anglické fráze Role-playing Game, do češtiny volně překládáno jako hra na hrdiny. Ve hrách tohoto typu hráč ovládá postavu či postavy v daném herním světě, prožívá a ovlivňuje jejich příběh. RPG jsou nejčastěji laděny do akčního či strategického stylu. Není však neobvyklé setkat se s logickými RPG.

V RPG je typický postupný vývoj postavy, vylepšování zbraní a schopností, které následně hráč využívá k boji proti nepřátelům. Ve hrách tohoto typu je nedílnou součástí interakce s nehráčskými postavami (non-playable character – NPC), které dokreslují celý příběh. RPG kladou důraz na atmosféru, kterou může podpořit zajímavý příběh, propracovaná grafika, tematická hudba nebo scény, kdy hráč pouze sleduje krátkou animaci, která mu dále představí vývoj dějové linky. Tyto animace jsou známy pod anglickým pojmem „cutscene".

RPG hry mohou být určeny jak pro jednoho hráče, tak i pro více hráčů. V tomto případě se hovoří o tak zvaných MMORPG (Massively Multiplayer Online Role-Playing Game). Stejně i zde hráč ovládá svou virtuální postavu, avšak nachází se ve stejném herním světě, jako ostatní lidští hráči. Ti mohou mezi sebou interagovat, bojovat, spolupracovat atd. Příkladem MMORPG může být hra Elder Scrolls.

Zde by bylo vhodné uvést rozdíl mezi frázemi „Multiplayer" a „Massively multiplayer". Multiplayer je hra po více hráčů, typicky pro dva (například piškvorky). Multiplayer hry mohou být hrány na jednom zařízení (lokální multiplayer) či online. Oproti tomu Massively multiplayer jsou hry pro násobně větší počet hráčů (v rámci stovek až tisíců), kteří mezi sebou mohou interagovat ve stejném online herním světě. MMO hry jsou na rozdíl od klasických Multiplayer her spuštěny i poté, co se daný hráč odhlásí.

Se hrami žánru RPG se lze setkat i mimo počítačové prostředí. Jako RPG se označují jakékoliv hry či akce, kdy se účastníci stylizují do role smyšlené postavy. Oblíbená je hra Dungeons and Dragons (do češtiny volně přeloženo jako Dračí doupě), kdy se sejde skupina lidí, jeden z nich vypráví příběh a ostatní hráči hrají postavy v tomto příběhu, interagují s předměty a jinými postavami ve smyšleném světě a snaží se splnit určený cíl – obdobně, jako to je u počítačových her tohoto typu.

Sportovní

Sportovní hry se mohou zařadit zároveň do jiných kategorií, nejčastěji mezi akční, jelikož vítězství zde ve většině případů tkví v rychlých reakcích. Jak již název napovídá, jedná se o hry simulující reálný sport jako například basketbal či hokej. Ve sportovních hrách se nachází postavy sportovců, které hráč ovládá. Herní pravidla jsou tvořena podle pravidel reálného sportu.

Za zmínku stojí nárůst virtuálních utkání mezi sportovními kluby během pandemie koronaviru. Fotbalisté se místo na hřišti sešli u herních obrazovek, prodali vstupenky fanouškům na tento virtuální zápas a fotbalový turnaj proběhl, i když reálné sportovní akce byly zrušeny (Petr Fojtík 2020).

Závodní

Zavodní hry obdobně jako sportovní spadají částečně mezi akční žánr. V závodních hrách hráč ovládá auto či obdobný prostředek, se kterým se účastní závodů. Cíl je v závodních hrách většinou jasný – dosáhnout konce jako první. Avšak i to nemusí být pravidlem, někdy musí hráč zajet dráhu ne rychle, ale například precizně bez žádné srážky se zdí. Závodní hry se mohou lišit různými aspekty, jako je například pojetí fyzikálních zákonů. Existují závodní hry, ve kterých se na fyziku příliš nedbá, auto hráče může jezdit po kolmých stěnách a podobně, což může vést k zajímavým a zábavným situacím. Dále se může do závodních her přidat prvek střelby, kdy hráč může například po sebrání určitého objektu vystřelit na jiného hráče, čímž ho zpomalí.

Prvek řízení auta se může vyskytovat i ve hrách, které by se nedali označit jako přímo závodní. Například hra Mafia patří mezi RPG, nicméně hráč se může přemisťovat pomocí dopravních prostředků, které sám ovládá. Jedna z dílčích misí v této hře je zúčastnit se automobilového závodu a vyhrát ho.

Příkladem čistě závodní hry je hra TrackMania.

Střílecí (First person shooter, Third person shooter)

Jedná se o typ akčních her s prvky střílení. Postava ovládaná hráčem disponuje střelnou zbraní. Střílecí hry lze rozdělit na hry s pohledem z první osoby (First Person Shooter – FPS), kdy je kamera umístěna v úrovni hlavy postavy a hráč se „dívá očima postavy“. Oproti tomu Third Person Shooter TPS označuje hry, kde je kamera umístěna za postavou (hráč

tedy vidí postavě do zad). Právě u tohoto typu hry jsou nejčastěji spatřovány prvky násilí, agresivity či krvavé scény.

Střílecí hry jsou populární hlavně na online platformách, kdy proti sobě soutěží více hráčů zároveň. Existuje velké množství různých přístupů, jak lze střílecí hra pojmout. Častým módem je tzv. battle royale, kdy hráči soupeří na uzavřené herní mapě, jejíž hranice se neustále zmenšují, čímž jsou všichni hráči soustředěni na jedno místo. Cílem hráčů je vyřadit soupeře a stát se posledním přeživším.

Dalším módem hodným zmínění je Competitive quest shooter, kdy je cílem hráčů plnit mise a úkoly, zatímco soupeří mezi sebou a mohou se vzájemně vyřadit – nevyřazený hráč s nejvyšším počtem splněných úkolů vyhrává. Toto tvoří zajímavou kombinaci, kdy hráč může zvolit více strategií, jak hru vyhrát – zaměří se na úkoly nebo vyřadí všechny hráče, kteří mají právě více bodů než on?

Příklad střílecí hry může být známá hra Counter-Strike.

Obrázek 2: Pohled z první a třetí osoby

Zdroj: (LevelCapGaming 2017)

S otevřeným světem (open world)

Ve hrách s otevřeným světem se hráči nachází v herním světě, kde je jim daná volnost tento svět objevovat, přetvářet ho a žít v něm. Oproti jiným hrám, kde je herní postup daný lineárně (nejprve začáteční úroveň, poté druhá a tímto způsobem až k poslední úrovni), hry v otevřeném světě nabízí hráči svobodu chodit po tomto světě bez větších omezení a plnit své vlastní cíle. Cíle hry mohou být nechány na hráči samotném – hráč si určí, zdali půjde splnit vedlejší úkol, postaví si dům, objeví novou planetu, anebo se vydá na cestu hlavního příběhu. Některé hry tohoto typu mohou mít pevně dané úrovně, kdy hráč nemůže vystoupit z lineárního postupu, avšak po skončení úrovně je hráč zas vrácen zpět do otevřeného světa.

U her s otevřeným světem se většinou jedná o akční RPG, nezřídka kdy se jedná o střílecí hry. Existují ale i závodní hry s otevřeným světem. Příkladem hry s otevřeným světem je Starbound.

Plošinové

Plošinová hra (anglicky platformer) je hra, ve které je cílem hráče překonat sérii překážek pomocí skákání, úhybů či boje. Plošinové hry se odehrávají buď ve 3D nebo ve 2D při pohledu z boku. Častým prvkem v těchto hrách jsou levitující plošiny, na které hráč může skočit, aby se dostal kupředu (odtud název plošinové hry). Příkladem 3D plošinové hry je Croc: Legend of Gobbos.

Arkádové hry

Jako arkádové hry se označují hry přístupné na herních automatech. Nezáleží na tom, jestli se jedná o akční, logickou, závodní či jakoukoliv jinou hru. Dnes se jedná spíše o historickou záležitost vzhledem k dostupnosti her na počítače či telefony. Původní arkádové hry je nyní možné hrát i na těchto zařízeních. Arkádové hry se vyznačují zastaralou grafikou (vzhledem ke své době) a často jednoduchými herními pravidly. Mezi arkádové hry patří například jedna z historicky prvních her Pong.

Kvízové / vědomostní / výukové hry

Posledním zmíněným typem her jsou kvízové hry. Jedná se zcela o jiný druh hry oproti výše zmíněným. Tento typ her cílí na vědomosti hráče. Ve hrách tohoto typu musí hráč zodpovídat vědomostní otázky. Hry často nemají žádný příběh – jedná se o pouhý test vědomostí. Díky hrám tohoto typu může hráč získat užitečné znalosti zábavnou formou. Tento typ her je oblíbený i v online prostředí, kdy hráči soupeří s ostatními ve svých znalostech. Příkladem vědomostní online hry může být česká hra Dobyvatel.

2.3 Dopad her na jedince

Lidský jedinec je neustále formován a ovlivňován vjemy z okolí – a to jak pozitivně, tak negativně. Pokud se člověk rozhodne trávit svůj čas hraním počítačových her, akceptuje fakt, že na něj bude tato hra jistým způsobem působit.

Průměrný hráč stráví hraním videoher 7,6 hodin týdně. Každý den se dobrovolně a se zapálením hráči věnují této aktivitě po více jak jednu hodinu. Počítačové hry tedy skýtají ohromný potenciál, který je možné využít či zneužít (Severin 2022).

Do dnešního dne bylo provedeno množství výzkumů zabývajících se dopadem videoher na jedince i na celou společnost. Počítačové hry jsou fenomén relativně nový, a ačkoliv se herní průmysl od historicky prvních pokusů o počítačové hry značně zdokonalil, lze očekávat stálý vývoj i do budoucnosti. Z tohoto důvodu je důležité zkoumat a popsat vliv her na dnešní svět. Pokud by se zjistilo, že negativní účinek převládá nad pozitivním, bude se vývoj počítačových her klonit jiným směrem a možná i upadat. Nebo naopak je možné odhalit, jaké aspekty her jsou nevhodné a měli by se odstranit a také podpořit ten typ her, který nejvíce přispívá pozitivnímu dopadu na společnost.

Je také nutno podotknout, že dopad hraní video her na individuálního člověka se může zásadně lišit v důsledku jiných faktorů, než je hra samotná. Hraje zde roli celkový stav jedince (jedná se o zdravého hráče či hráče s psychickou poruchou), jeho motivace k hraní (jedná se o závislost či jen o příležitost setkat se s přáteli při příjemné zábavě), sociální interakce během hry (během hraní se hráč setká s vulgárním oponentem či nalezne přátelského protihráče) a další. (Halbrook et al. 2019)

2.3.1 Pozitivní dopad

V této kapitole budou popsány vlivy počítačových her, které se dají považovat za prospěšné. Budou zmíněny jen ty z nich, na kterých se odborníci obecně shodují. Mnohé studie a pokusy totiž dokazují protichůdné výsledky, avšak na některých aspektech je možné se shodnout. (Melissa T. Buelow et al. 2015)

Také je nutné rozdělit hry na edukativní, které přímo cílí na rozvoj člověka, a na nevýukové, kde je hlavním cílem zábava a odreagování. Nicméně u těchto her lze také zpozorovat jisté pozitivní efekty na hráče.

Edukativní hry

Záměrem výukových her je primárně předat užitečné informace či simulovat situace, kde se žák naučí novým dovednostem. Kombinace s herními prvky hráče motivuje ve hře vytrvat a učit se tím pádem déle. Rozdíl mezi běžnými a edukativními hrami tkví v jejich záměru. U

běžných her je cílem hráči nabídnout zábavu či relaxaci a jako vedlejší efekt se může něco nového naučit. Primárním cílem edukativních her je naopak předat hráči informace či ho zlepšit v různých dovednostech a mimo jiné to daného hráče může bavit.

Lze začít citátem „Video hry nejsou nepřítelem, ale naší nejlepší příležitostí, jak zapojit děti do učení“ (Prensky 2003, volně přeloženo). Jak již bylo zmíněno, počítačové hry skýtají široký potenciál. Pokud tento potenciál bude využit k motivaci žáků v učení, mohou žáci sami sebe vzdělávat, aniž by si to sami uvědomovali. Výukové hry mohou být mimo jiné využity jako edukativní pomůcka pro učitele. Je jistě vhodné zmínit online kvízovou hru Kahoot!, která se stala běžnou součástí vyučovacích hodin během distanční výuky, avšak i po skončení pandemie ji učitelé s oblibou používají na ozvláštnění svých hodin.

Mezi edukativní hry mohou patřit i simulační hry. Již v roce 1946 na MIT vznikl simulátor na řízení letadla. Od té doby se technologie značně vyvinuly. V dnešní době, kdy jsou rozšířené technologie jako například virtuální realita, je možné simulovat operaci člověka, stavbu domu či jiné těžko dostupné situace. (Noemí a Máximo 2014)

Nevýukové hry

Druhá podkapitola patří hrám, jejichž cíl není vzdělávání hráče. Jedná se tedy o hry, které hráče formují nepřímo. Avšak i v tomto případě je popsáno množství pozitivních dopadů.

První z pozorovaných dopadů na schopnosti hráče je zvýšení prostorově vizuální schopnosti (jako je prostorová představivost, orientace v prostoru atd.). Bylo zjištěno, že krátkodobé hraní video her nemá na tyto kognitivní schopnosti zásadní vliv, zatímco u pravidelných hráčů byl v prostorově vizuálních schopnostech zaznamenán vývoj. (Greenfield et al. 1994)

Další efekt, který je vhodný zmínit, je zrychlení reakční doby. Toto i intuitivně dává smysl, jelikož u drtivé většiny akčních her jsou klíčové rychlé a přesné reakce, tudíž je hráč motivovaný zlepšovat svou reakční dobu. Existují dokonce i tréninkové simulátory na zrychlení a zpřesnění reakcí. Hráči tímto způsobem mohou plynule přejít k cílenému zlepšování svých schopností.

Mezi další pozitivní důsledky hraní her na kognitivní funkce patří zlepšení inhibiční kontroly (tj. potlačení impulzivních reakcí a zvolení vhodnějšího chování k dané situaci), abstraktního uvažování, pracovní paměti a schopností při řešení problémů.

Co se týče psychického stavu, byla zjištěna pozitivní korelace mezi hraním her pro více hráčů a psychického zdraví. Toto platí jak pro kompetitivní hry (kdy dva či více hráčů hrají proti sobě), tak i pro kooperativní hry (kdy se více hráčů snaží spolupracovat za dosažením výhry). U kooperativních her byl zjištěn ještě silnější pozitivní dopad než u kompetitivních, u kterých často přicházejí emoce jako je vztek, pomstychtivost či agrese. Citovaná studie (Halbrook et al. 2019) také sděluje, že se nemusí jednat pouze o kooperaci s lidským hráčem, nýbrž i hry, ve kterých je implementována interakce mezi hráčem a nehráčskými postavami kontrolovanými počítačem, mohou vést ke zlepšení sociálních schopností hráčů.

Dále bude poukázáno na fyzické zdraví. Na trhu existují aplikace, které kombinují herní prostředí a fyzickou aktivitu. Takovéto hry jsou většinou ovládány senzory, které reagují na fyzický pohyb hráče. Jako příklady takových her mohou být uvedeny Just Dance či Beat Saber. Just Dance je hra, kdy se hráč snaží napodobovat tanec podle animované postavy. Následně je snímán jeho pohyb, na jehož základě je hráč ohodnocen počtem hvězd, což ho motivuje se stále zlepšovat a dosáhnout maximálního skóre. Druhá hra Beat Saber je hra pro virtuální realitu, kdy se hráč snaží virtuálními světelnými meči zasáhnout objekty ve hře, zatímco se musí skrčovat a uhýbat před překážkami. Hry ve virtuální realitě téměř vždy vyžadují alespoň minimální fyzickou aktivitu, často se však jedná i o náročnější pohyby, jako je boxování, skákání, úhyby a podobně.

Hry cílené na pohyb ukazují obdobné benefity jako tradiční cvičení. Navíc oproti klasickému cvičení prokazují video hry větší vytrvalost člověka setrvat v pravidelném cvičení i nadále, což je právě přikládáno provázanosti s herními prvky. (Halbrook et al. 2019).

Jako další jsou zde zmíněny benefity hraní her na vzdělání (stále se nejedná o výukové hry). Skrze hraní her může hráč získat tytéž znalosti jako při učení ve škole či z knih, nicméně zde je motivovaný hrou samotnou. Různá témata z dějepisu lze obsáhnout hraním her založených na historii, např. Medieval: Total War™. Taktéž vzdělání v oblasti zeměpisu může být podpořeno hrami (většinou strategickými), jejichž herní mapa se shoduje s mapou světa, např. Europa Universalis IV. Obdobně lze nalézt hry s prvky chemie, biologie, fyziky, ekonomie atd.

Mezi neodmyslitelné benefity hraní počítačových her je zdokonalování cizích jazyků. V případě, že člověk začne hrát hru v jiném než rodném jazyce (což u českých hráčů znamená drtivou většinu her), je silně motivovaný rozumět tomu, co se ve hře odehrává. O

to spíše, pokud se jedná o kooperativní hry pro více hráčů, kde je zásadní komunikace mezi jednotlivými hráči. Mnoho hráčů se tímto způsobem může zdokonalit v cizím jazyce, nebo se dokonce nový jazyk naučit od nuly jen skrze počítačové hry.

Na závěr budou uvedeny logické hry, kde je benefit očividný. Existuje množství her, které jsou přímo zaměřeny na logické myšlení. Hráčům se tímto způsobem zábavnou formou zlepšuje logické uvažování. Mezi zástupce logických her může patřit česká hra Berušky II, kde je cílem hráče přesouvat objekty ve 3D prostoru za účelem dosáhnout východu. Logické problémy lze často shledat i v jiných typech her. Například hra Trine je převážně akčního typu, nicméně k postupu dál musí hráči vyřešit různorodé logické překážky.

2.3.2 Negativní dopad

V předchozí kapitole byl popsán pozitivní dopad her na jedince i na hráče, avšak i zde lze nalézt stinnou stránku. Zatímco hry mohou přinést odpočinek, zábavu, podpořit socializační schopnosti hráče či ho naučit novým věcem a zdokonalit v jeho dovednostech, mohou videohry představovat i značné riziko. Hraní počítačových her může poškodit fyzický i psychický stav člověka. Hráč si snadno na hrách vybuduje závislost, což negativně ovlivní jeho sociální život.

S počítačovými hrami se běžně mohou setkat děti ještě dříve, než se vůbec naučí mluvit. Jak je obecně známo, různé podněty mohou mít na vývoj dítěte mnohem větší dopad než na dospělého. A to samozřejmě platí i u počítačových her. Z tohoto důvodu jsou potenciálně nevhodné hry omezeny věkem. Jedná se především o hry obsahující násilí, hazard, erotické či vulgární prvky. Je však nutné podotknout, že uvedení věkové hranice na úvodním obrázku hry nezabrání mladším dětem hru spustit. Je překvapující, že velké procento rodičů se neřídí věkovým doporučením a povolí dětem hrát hry označené jako nevhodné pro nezletilé. Výsledky průzkumu ukazují, že až 86 % z více než dvou tisíc tázaných rodičů povolí dětem hrát hry pro dospělé (zatímco u nevhodných filmů je to jen 23 %). Více než čtyři pětiny rodičů si nemyslelo, že hraní her pro dospělé může mít negativní vliv na jejich dítě, avšak 43 % rodičů zpozorovalo zhoršení chování dítěte a 48 % přiznalo, že jejich dítě je pravděpodobně na hře závislé. Je tedy zarážející, že sledování nevhodných filmů rodiče zakazují, zatímco u počítačových her jsou povolnější. (Childcare.co.uk 2023)

Počítačové hry pro dospělé obsahují prvky, kterým mladší děti nemohou plně rozumět (sexuální scény apod.), což následně může narušit jejich psychický vývoj. V mladším věku mají děti navíc tendenci napodobovat pozorované chování. Pokud tedy hrají hru s vulgárními a agresivními scénami, mohou se „inspirovat" a chovat se taktéž. V případě online hry pro více hráčů není výjimkou setkat se s toxickým chováním ze stran ostatních hráčů. Desetileté dítě se tak může setkat s kyberšikanou již v takto útlém věku. (O'Shea 2015)

Nyní bude pozornost přenesena na negativní účinky ve hraní her při lepším scénáři, kdy hráči poslechnou doporučení výrobců hry (věková hranice, doporučená doba hraní atd.). Prvně lze znovu zmínit násilí ve hře. Průzkum dokázal, že hraní násilných her může vést k nižší empatii, snížené citlivosti na násilí a umírnění prosociálního chování. Podle odborníků stačí jen krátkodobé vystavení k násilím ve hře na to, aby se u hráče projevilo agresivní chování, nebo alespoň myšlenky na násilí. Další studie ukazují, že existuje pozitivní korelace mezi hraním násilných her a vstupování do fyzických soubojů v reálném světě (Jordan a Romer 2014).

Další běžně se objevující dopady přílišného hraní video her je schopnost udržet pozornost. Tento fenomén je obecně známý a studie ho jen potvrzují. Herní prostředí je plné silných stimulů, které neustále zachycují hráčovu pozornost, ať už vizuálně (záblesky, výbuchy…) či zvukově (střelba, křik…). Reálný svět je mnohem pomalejší a skýtá méně podnětů než počítačové hry. Tím pádem v situacích, které ze své podstaty nepřitahují lidskou pozornost (např. výklad učitele), mohou pravidelní hráči pozornost ztrácet. (Groves a Anderson 2017)

Existuje celá řada studií zkoumající negativní vliv na hráče video her, které je nemožné obsáhnout do jedné podkapitoly. Dále zde budou jen vyjmenovány další prokázané dopady, které negativně ovlivňují hráče. Patří mezi ně: zbytečné vstupování do riskantních situací například při řízení (Fischer et al. 2007), zvýšená náchylnost k epilepsii (Bureau et al. 2004), zhoršená kvalita spánku, vyčerpanost, obezita, deprese, problémy se srdcem, špatný pitný a jídelní režim, nedostatečná motivace k jiným aktivitám než je hraní, závislost na hrách, sebevražedné myšlenky nebo chudý vnitřní svět člověka (Adair 2021, podpořeno vlastním pozorováním)

2.4 Současné trendy

Rozdělení hráčů podle pohlaví

Z běžného pozorování herních komunit by se dalo usoudit, že většina hráčů jsou muži. Ačkoliv se to může zdát překvapivé, procento žen hrajících pravidelné hry značně vzrostlo a téměř se s muži srovnalo. Dle výzkumu v USA tvoří ženy mezi hráči 48 % (Entertainment Software Association 2022). Zde je nutné stanovit, kdo je počítán mezi hráče. Ve veřejných diskuzích lze najít příspěvky tvrdící, že pokud bychom uvažovali i hráče mobilních her, mohou ženy tvořit až 65 %. Různé zdroje navrhují, že poměr žen a mužů se může značně lišit v různých žánrech her (Erin Hamilton 2009). Zatímco v brutálních hrách je stále drtivá část mužská, ženy zas dominují ve hrách s prvky sociálních interakcí či v příběhových hrách.

Jeden z důvodů, proč se ženy příliš neobjevují v komunitě online střílecích her, může být vysoká toxicita přímo vůči ženám. Z průzkumu (Assunção 2016) bylo zjištěno, že při projevení ženské identity dochází k urážkám a hostilnímu chování ze strany ostatních hráčů, což přirozeně ženy demotivuje hrát danou hru dále. Z 241 tázaných žen 150 potvrdilo toxické narážky ze strany herní komunity během hraní her.

Jako další důvod navržený na různých fórech (se kterým se autor ztotožňuje) může být vliv rozdílů mezi ženským a mužským chováním. Ve hrách, kde je možné projevit dominanci, agresivitu a podobné vlastnosti typicky spojované s muži, není překvapením, že většinové zastoupení budou tvořit právě muži.

Na závěr této podkapitoly každopádně vyvstává otázka, jak se vývojáři postaví k faktu, že poměr žen a mužů hrajících jejich hry dosahuje jedna ku jedné. Zároveň se však musí postavit tomu, že v herním prostředí dochází k hostilním projevům vůči ženám. Například vývojáři hry Cyberpunk 2077 zrušili volbu pohlaví při tvorbě avataru postavy (Greene 2019). Zde je necháno na čtenáři, aby si sám vytvořil názor, zdali to je krok správným směrem či nikoli.

Peníze ve videoherním průmyslu

Vývoj kvalitní hry vyžaduje spoustu času, úsilí a peněz. Například vývoj hry Witcher 3: Wild hunt trval tři a půl roku týmu 150 zaměstnanců. Společnost celkem za tuto hru zaplatila přes 80 milionů dolarů (Burke 2015). Hlavní zdroj příjmu tvoří prodané kopie hry. Další

možný způsob, jak vývojáři mohou získat výdělek ze své hry, je skrze placené reklamy, čehož využívají převážně mobilní hry. Další příjem činí nákupy virtuálního zboží ve hrách samotných (vzhled postavy, silnější zbraně apod.). Velký vzrůst peněz v herním průmysl nastal během pandemie kovidu. Dnes se hráčům nabízí nové možnosti, jak utratit své peníze, například pořízením zařízení na virtuální realitu, což v dřívějších dobách nebylo tak časté.

Co se týče konkrétních cifer, k roku 2023 hodnota celkového videoherního průmyslu stoupá k 217 bilionům dolarů, což činí nárůst o 10 % oproti minulému roku (Jessica Clement 2022). Průměrný hráč utratí měsíčně v přepočtu přes 1500 Kč za nové hry, nákupy virtuálního zboží či za předplatné. (Catherine Lewis 2022)

Popularita herních žánrů

Je přirozené, že některé videoherní žánry jsou oblíbenější než jiné. Konkrétní hodnoty se liší podle věkových skupin či podle pohlaví, avšak i přesto lze učinit přibližné závěry. Podle statistiky k roku 2022 v oblíbenosti vítězí střílecí hry, těsně následují akční dobrodružné hry. Dále to jsou simulační hry, sportovní a závodní. Teprve až poté se řadí strategické hry, další v pořadí jsou logické hry. Na předposledním místě jsou plošinové hry, a nakonec online deskové hry, které jsou oblíbené převážně jen u starších hráčů. (Jessica Clement 2023)

3 Vývoj online her

Vytvořit kvalitní online hru vyžaduje překonat velké množství problémů, které budou popsány a vysvětleny v následujících podkapitolách. Samozřejmě ve velké míře záleží na samotném typu online hry. Strategická hra, kde se hráči střídají po tazích, bude tvořena pomocí odlišných algoritmů než závodní hra, kde hráči zápasí v reálném čase.

Jelikož by bylo nemožné popsat všechny aspekty všech typů online her, je tato práce zaměřena na akční online hry. Samozřejmě je stále nereálné zmínit naprosto vše, co vývojář potřebuje ke zdárnému dokončení online hry, nicméně hlavní body, se kterými je možné se setkat ve většině her tohoto typu, jsou níže popsány.

3.1 Herní architektura a algoritmy hry

V této podkapitole jsou popsány problémy týkající se logiky hry bez online komunikace, která bude popsána později. Zatím zde budou popsány algoritmy, které by byly využity i v offline hrách pro jednoho hráče. Pohyb hráče, kontrolování jeho akcí, interakce mezi objekty – to vše je potřeba vyřešit jak u online, tak i u offline her.

3.1.1 Pohyb objektů

V typické akční online hře předpokládáme, že objekty (jako jsou střely, hráč, nepřátelé atd.) se mohou pohybovat a interagovat mezi sebou. Tento pohyb je samozřejmě potřeba řídit a kontrolovat. Existuje více přístupů, níže budou popsány některé z nich.

Nejjednodušší, avšak velmi rigidní přístup pro naprogramování pohybu objektů je realizovat ho pomocí absolutního měnění souřadnic objektu. Pokud je například hráč na pozici [0; 10] a stiskne šipku nahoru, změní se jeho souřadnice na [0; 11]. Vykreslí se nová pozice postavy, postava se zastaví a tím je pohyb u konce.

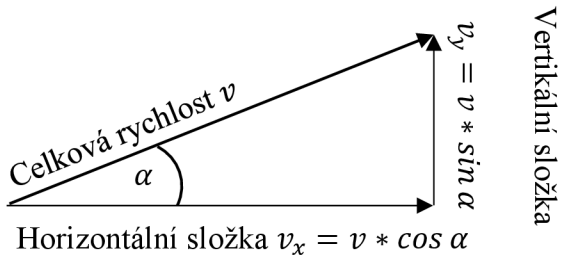
S tímto přístupem vystačí však jen velmi primitivní hry, které nepředpokládají jakýkoliv složitější pohyb. Již u velmi jednoduchých požadavků je tento přístup nedokonalý. Pokud by vývojář chtěl naprogramovat pohyb bližší realitě, postupoval by pravděpodobně tak, že

při stisknu klávesy hráč nejprve zrychlí, následně půjde konstantní rychlostí, a nakonec, když hráč klávesu pustí, nezastaví okamžitě, ale postupně se zpomalí.

Zde by bylo vhodné pracovat s měnící se rychlostí. Každý objekt, u kterého se předpokládá pohyb, bude mít rychlost ve vodorovném a svislém směru. V každém snímku se změní jeho souřadnice osy x o vodorovnou rychlost a souřadnice y o rychlost svislou. Když objekt začne svůj pohyb, může se rychlost postupně zvyšovat a zas při ukončení pohybu postupně zpomalovat. U tohoto přístupu je dále důležité stanovit, jakou souřadnou soustavu zvolit. Nabízejí se dva přístupy – kartézské souřadnice či polární souřadnice.

Kartézské řeší pohyb v kolmých osách x a y (případně z pro 3D). Je potřeba znát velikost horizontální a vertikální složky rychlosti. Pokud se hráč pohne vlevo, změní se souřadnice x. Pokud nahoru, jedná se o souřadnici y. Výhoda tohoto přístupu je, že vykreslování na obrazovku je ve většině případů realizováno pomocí kartézských souřadnic, což zde přímo nahrává této architektuře. Objekt se pohybuje po ose x a y, jeho souřadnice jsou přímo předány pro vykreslení a na monitoru se zobrazí daný objekt.

Druhý přístup řeší pohyb pomocí úhlu a velikosti rychlosti. Tento návrh je vhodný například v situaci, kdy předpokládáme časté otáčení objektů. Otočí-li se hráč, změní se úhel. Pokud se hráč pohne kupředu, je jeho souřadnice x změněna o součin velikosti rychlosti a cosinus daného úhlu, o který je otočen. Obdobně to platí pro vertikální směr, zde je však vynásobena rychlost sinem úhlu.



Obrázek 3: Složky rychlosti

Tímto způsobem je možné lehce převádět kartézské a polární souřadnice podle potřeby.

Návrh pohybu je flexibilnější, ale stále neřeší složitější situace. Výsledná rychlost hráče může být skládána z více složek. Hráč se pohybuje kupředu, zároveň se ve hře vyskytne prvek větru, který bude hráče posouvat jedním směrem a k tomu navíc soupeř vytvořil přitažlivé pole, které hráče vtahuje do svého středu. Vývojář zde musí řešit tři složky zároveň – pohyb, vítr a pole. Pohyb hráče sám o sobě byl popsán výše – postupné zrychlení při počátku a zpomalení při ukončení pohybu. Vítr bude neustále posouvat hráče konstantní rychlostí směrem doprava. Nakonec přitažlivé pole působí tak, že hráč neustále nabírá zrychlení směrem ke středu pole. Řešit tuto složitou situaci, ke které potenciálně může dojít, by bylo přes pouhé rychlosti zbytečně složité. Výhodnější řešení je rozdělit si pohyby samostatně. Ke každému objektu bude možné přiřadit více pohybů. Každý pohyb bude mít své vlastnosti (směr, velikost, míra zrychlování či zpomalování aj.). V každém snímku hry budou rychlosti všech pohybů sečteny. Objekt se pohne ve směru výsledné rychlosti. Následně budou provedeny případné změny všech pohybů (směr síly přitažlivého pole se mírně změní, vítr může zesílit, hráč se přestává pohybovat, takže rychlost jeho pohybu se bude zmenšovat). Tento návrh je dostatečně flexibilní, aby bylo možno začlenit jakýkoliv nový pohyb.

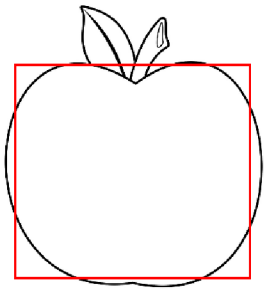
3.1.2 Detekce kolize

V průběhu typické akční hry je třeba neustále kontrolovat, zdali se nějaký objekt nesetkal s jiným. Ať už se jedná o dotyk postavy se zdí, kolize výstřelu s nepřítelem nebo náraz dvou hráčů – vše je potřeba zaznamenat a vyhodnotit. V následujících kapitolách budou popsány různé přístupy, které usnadňují detekovat srážky objektů, a nakonec i samotné algoritmy pro řešení tohoto problému.

Obalový objem (bounding volume)

Prvně je potřeba určit, co vlastně bude kontrolováno. Každému objektu ve hře lze přiřadit souřadnice X a Y (případně Z pro hru ve 3D). Základní myšlenka pro detekci kolize je kontrolovat souřadnice objektů a jejich rozměry. Pokud se dva objekty dotýkají či překrývají, lze situaci považovat za kolizi těchto dvou objektů. Nicméně zde nastává problém s tím, jak přesně definovat pojem „rozměr objektu“. V reálném světě není de facto nic perfektní geometrický tvar – míč je přibližně koule, krabice je přibližně kvádr. Nějaká tělesa však nelze vyjádřit geometrickým tvarem. Například člověka, strom či vidličku nelze vyjádřit konkrétním geometrickým tělesem.

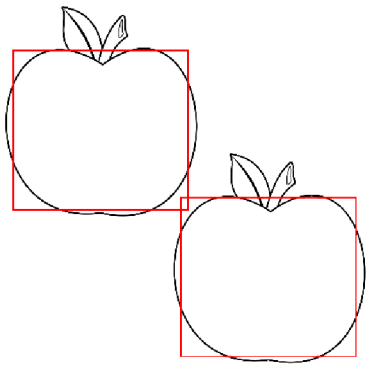
Z tohoto důvodu je vhodné rozměry těles aproximovat na obdélník (či kvádr pro tři dimenze. Dále budou popisovány situace pro dvě dimenze, nicméně ty samé algoritmy lze užít i pro třídimenzionální hru.). Rozměr obdélníku je určen výškou a šířkou a jeho pozice určena dvěma čísly X a Y. Určit kolizi dvou obdélníků je již triviální. Pokud tedy veškeré objekty ve hře budou aproximovány na obdélníky, je určení jejich kolizí otázkou porovnání souřadnic a velikosti obdélníků. Nejedná se o vzhled objektu, nýbrž o jeho rozměry, které program bude uvažovat ve svých algoritmech. Tento uvažovaný tvar objektu se nazývá obalový obdélník (anglicky bounding box). Toto lze demonstrovat na příkladu objektu ve tvaru jablka.



Obrázek 4: Obalový kvádr

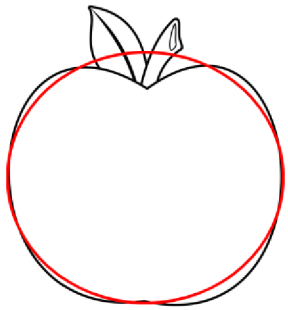
Zdroj: Rheo 2015, upraveno (tak i u následujících)

Na předchozím obrázku lze zpozorovat chybu na krajích objektu. Ačkoliv obdélník většinu své plochy sdílí s původním obrázkem, je zřetelné, že na rozích obalový obdélník přesahuje obrázek, a naopak uprostřed stran obrázek přečnívá. Vzhledem k této nepřesnosti se může stát, že u dvou objektů je vyhodnocena kolize, aniž by se vizuelně překrývaly.



Obrázek 5: Překrývající se obalové objemy

Užití obdélníku tedy není vždy nejvhodnější. V momentě, kdy jsou objekty jiného tvaru než obdélníku, lze pro obalový objem užít jiné základní geometrické obrazce, než je obdélník. Pro zaoblené objekty se nabízí kruh. Na následujícím obrázku lze zpozorovat, že problém s přečnívajícími rohy, jako to bylo ve verzi s obdélníkem, naprosto zmizel a nepřesnosti jsou zcela zanedbatelné.



Obrázek 6: Eliptické ohraničení

Při použití kruhu (nebo elipsy pro ještě přesnější ohraničení) stačí pro detekci kolize porovnat souřadnice obou objektů a jejich poloměr. Užití jiných obrazců je samozřejmě možné. Zde si vývojář volí mezi přesností obalu a výpočetní náročností. Mezi běžně užívané tvary patří otočený obdélník nebo více úhelník.

Lze si jistě představit situace, kde by byla přibližná aproximace objektu obdélníkem či elipsou neakceptovatelná. Mezi takové případy patří například hry, které se snaží velmi přesně simulovat fyzikální děje. Velmi často se jedná o logické hry, kde veškerý pohyb těles a jejich srážky je nutné vyhodnocovat s velkou přesností (jako například ve hře Brain it on).

V této situaci stojí volba na vývojáři. Buďto se spokojí s menšími chybami za cenu rychlého algoritmu, nebo si vybere složitější přístup. Ve většině online her je dána přednost rychlosti, avšak stojí za to se zmínit i o algoritmech pro naprostou přesnost.

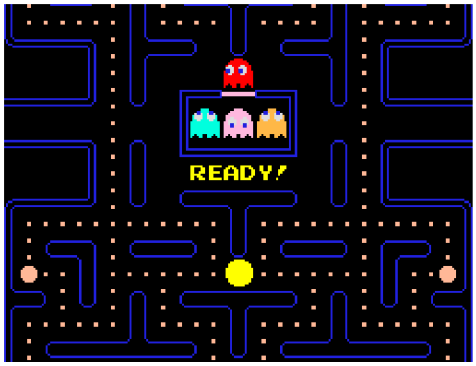
V první řadě je třeba jakýmkoliv způsobem reprezentovat přesný tvar objektu. Jedna možnost je uložit tvar pixel po pixelu do dvourozměrného pole. Druhá možnost je reprezentovat objekt pomocí více jednoduchých geometrických tvarů (na způsob vektorové grafiky).

V momentě, kdy se protne obalový objem jednoho objektu s druhým, není kolize rovnou vyhodnocena, pouze se zaznamená, že se o kolizi jednat může. Následně se porovnají přesné

tvary obou objektů a bude orčeno, zdali se opravdu protnuly, či nikoliv. Pokud ano, je srážka vyhodnocena, v opačném případě se v dalším snímku hry, kdy se objekty zas pohnuly, zkontroluje, zdali se neprotnuly tentokrát a toto se opakuje do té doby, dokud se opravdu nesrazí nebo dostatečně nevzdálí.

Tento přístup je však výpočetně náročný ve srovnání s použitím pouhého obalového objemu, a proto se vývojáři online her spíše kloní k jednodušším algoritmům.

Na závěr této podkapitoly je vhodné zmínit i případ, kdy je použití obalového objemu zbytečné. Jedná se o hry, kde jsou pozice jednotlivých objektů uspořádány do pevně dané mřížky, jako to je například ve hře PacMan nebo při počítačové verzi šachů. V tomto případě je velikost objektů jedno pole mřížky. V momentě, kdy postava přechází z jednoho pole do sousedního, může být uvažováno, že se nachází v původním i sousedním poli zároveň. Detekce kolize je v tomto případě triviální. V případě pohybu je kontrolováno, co se nachází na vedlejším poli, kam pohyb směřuje. Pokud se tam nachází jakýkoliv objekt, je vyhodnocena kolize. (Ericson 2004)



Obrázek 7: PacMan – příklad hry v mřížce

Zdroj: (Namco Limited 1980)

Posteriori, a priori

Další problém, o kterém je vhodné se zmínit, se týká momentu vyhodnocení kolize. V předchozí podkapitole o obalovém objemu byla kolize uvažována v momentě, kdy se obalový objem obou objektů protínal. Nicméně při porovnání s reálným světem je toto nesmysl. Pokud si například člověk stoupne na zem, jeho noha se zastaví v momentě dotyku s podlahou, a ne až poté, co je částečně v podlaze zanořený. Stejně tak i ve hrách by srážka měla být vyhodnocena před tím, než se objekty protnou. V některých případech je toto irelevantní. Pokud se jedná o letící střelu, která je při dotyku s hráčem zničena, nehraje roli,

pokud je kolize vyhodnocena až poté, co se střela protíná s hráčem. Avšak při dotyku se zdí je většinou žádoucí, aby se postava do zdi nezabořila.

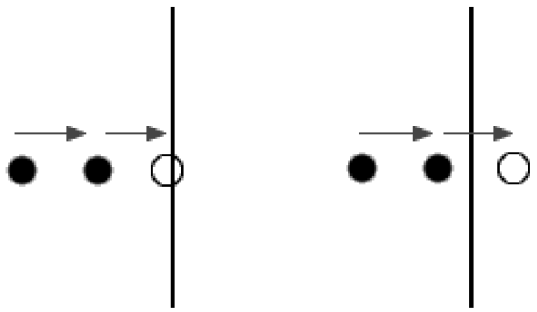
Tyto dva přístupy se nazývají „a priori“ a „posteriori“. Posteriori je ten případ, kdy se objekty protnou a následně je až vyhodnocena kolize. A priori naopak vyhodnotí kolizi ještě před tím, než se objekty setkají.

V obou případech je většinou vhodné upravit přesné souřadnice. V případě posteriori, kdy se objekty již protnuly, je žádoucí upravit pozici tak, aby se těsně dotýkaly. V případě a priori jsou zas objekty stále vzdáleny od sebe, tudíž je vhodné přepsat souřadnice k přesnému dotyku.

Detekce srážky při velkých rychlostech

Pohyb tělesa je v základu prováděn tak, že z původních souřadnic se skokově přemístí na souřadnice nové na základě směru a velikosti jeho rychlosti. Před tím (anebo poté, což závisí na přístupu a priori či posteriori) se zkontroluje, zdali nedošlo ke srážce s jiným objektem. Tím, že se scéna mění přibližně šedesátkrát za sekundu, lidský mozek vyhodnotí pohyb jako plynulý.

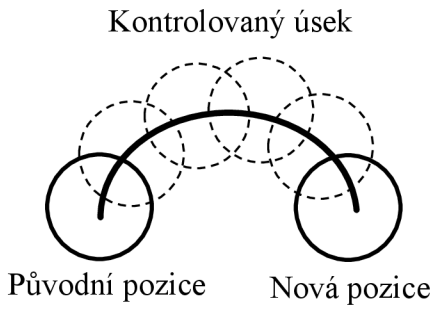
Avšak v momentě, kdy je rychlost určitého objektu příliš vysoká v poměru k jeho rozměrům, může dojít k problému. Rychle pohybující se tělesa jsou v online akčních hrách běžné – většinou se jedná o střely. Jak již bylo řečeno, objekt je při pohybu skokově přemístěn na novou pozici v závislosti na rychlosti. Pokud je rychlost příliš vysoká, může se stát, že vzdálenost skoku mezi starou a novou pozicí je vyšší než šířka tělesa. Tím pádem může nastat situace, kdy rychlý objekt přeskočí jiný předmět, ačkoliv by se s ním srazit měl. Tento přístup se nazývá diskrétní detekce kolize.



Obrázek 8: Problém rychlého pohybu

Sekvence tří snímků: nalevo se objekt srazí se zábranou, avšak napravo objekt zábranu přeskočí.

Problém může být vyřešen velmi snadně. V první řadě program zkontroluje, zdali se objekt pohybuje vyšší rychlostí, než je jeho šířka. Pokud ano, postačí zkontrolovat všechna místa, kde se objekt nacházel, mezi starou a novou pozicí. Tento přístup se nazývá kontinuální detekce kolize. Tímto způsobem se již nemůže stát, že by byl jakýkoliv objekt ležící v cestě ignorován. Důležité je nekontrolovat místa ležící na úsečce mezi původní a novou pozicí, ale v případě nepřímočarého pohybu na místech, kde by se reálně objekt nacházel. (Čejka 2016).



Obrázek 9: Kontinuální detekce pro nepřímočarý pohyb

Každý s každým

Tato podkapitola (včetně následujících) bude věnována řešení toho, mezi jakými objekty kolize bude vyhodnocována. Nejprimitivnější přístup je kontrolovat neustále možnou kolizi každého objektu s každým jiným porovnáváním jejich souřadnic a rozměrů. Tento algoritmus je nejsnadnější na implementaci, avšak časová náročnost je ve většině případů neúnosná. Časová náročnost tohoto algoritmu je $O(n^2)$. Uvažuje-li se akční online hra s prvkem střílení, může být počet objektů (hráči, počítačoví nepřátelé a hlavně střely) v řádu stovek či tisíců. Již pro sto objektů by byl počet nutných operací přibližně deset tisíc. Detekce kolize je prováděna každý snímek a je to jen jeden z mnoha algoritmů, které je nutno ve snímku vykonat. Z tohoto důvodu jsou vyvinuty vhodnější algoritmy pro řešení tohoto problému, kterou jsou popsány níže.

Snadné optimalizace

Lze implementovat snadné optimalizace, které ušetří několik málo operací, a to jak u algoritmu „každý s každým“ nebo i u následně popsaných. Ačkoliv vyhodnocování každého tělesa s každým jiným stále nebude pravděpodobně zrychleno na přijatelnou úroveň, stále stojí za zvážení tyto optimalizace přidat.

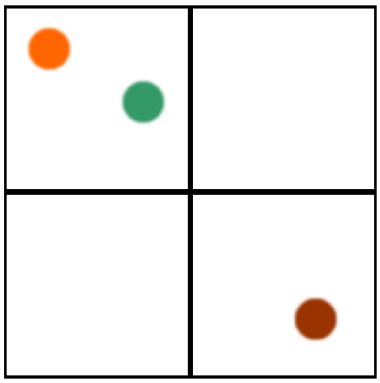
První návrh je nevyhodnocovat možné kolize u objektů, které se nepohybují. V případě, že se dva objekty vůči sobě navzájem nachází v klidu, nemohou se srazit. Do nepohybujících se těles může narazit pouze jiný objekt v pohybu.

Druhá myšlenka stojí na faktu, že jedna dvojice objektů stačí být zkontrolována jen jednou. Pokud se provedla kontrola kolize objektu A s objektem B, nemusí se kontrolovat srážka objektu B s objektem A, protože výsledek této kontroly již známe.

Nakonec je také zbytečné kontrolovat srážku objektu se sebou samým.

Quadtree

Další možné zlepšení tkví v tom neprovádět detekci kolize u objektů natolik vzdálených, že by se v daném snímku reálně setkat nemohly. Tento algoritmus je implementován rozdělením prostoru herního pole na části. V prvním kroku je prostor rozdělen na čtyři stejné sekce. Následně při detekci kolize stačí, když budou kontrolovány dvojice objektů nacházející se pouze ve stejné sekci.



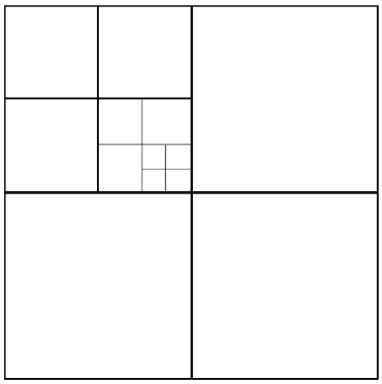
Obrázek 10: Demonstrace quadtree

Žlutý objekt bude kontrolován pouze se zeleným, a ne s hnědým.
Hnědý se totiž nachází v jiné sekci.

Pro každou sekci existuje seznam všech objektů, které se tam právě nacházejí a pro které bude prováděna detekce kolize (zde již každý s každým). V případě, že se objekt pohne z jedné sekce do druhé, bude odebrán ze seznamu původní sekce a přidán do nové.

Rozdělení prostoru na čtyři sekce sice pomůže, ale pro velké množství objektů by nedošlo k příliš velkému zlepšení. Intuitivně se nabízí rozdělit sekci na další čtyři. A tu v případě potřeby na další čtyři. Každá sekce může být dělena teoreticky do nekonečna. Hranice, kdy přestat dělit, může být brána prostorově (pevně určeno – nejnižší sekce má velikost 100 x

100 pixelů) nebo podle počtu objektů (pro méně jak 5 objektů není potřeba dělit dále). Tímto způsobem je dosažena značná optimalizace, kdy z původní složitosti $O(n^2)$ pro kontrolu každého s každým je algoritmus zlepšen na složitost O(n log n). Pro tisíc objektů není potřeba provádět milion operací, ale pouze tři tisíce.



Obrázek 11: Větvený quadtree

Nastává zde samozřejmě několik menších problémů. V první řadě celý quadtree je třeba neustále měnit v závislosti na pohybu objektů. Toto samozřejmě zabere jistou dobu, nicméně optimalizace, kterou quadtree nabízí, stále převáží několik málo příkazů pro modifikování této struktury.

Následující problém nastává v momentě, kdy se objekt nachází na rozhraní dvou buněk. V tomto případě je nutné uvažovat jeho přítomnost jak v jedné, tak i v druhé buňce (protože teoreticky může kolidovat s ostatními objekty v obou buňkách).

Tento přístup lze samozřejmě použít i pro prostředí ve třech dimenzích. Zde místo čtverců dělíme krychle na osm menších krychlí. Zde se používá označení octree.

Ve třech dimenzích ještě nastává problém samotného ohraničení prostoru. Hranice herního pole pro osu x a pro osu y jsou většinou dané. Pokud má hra připomínat reálný svět, spatříme na okrajích nepřekročitelné hranice – většinou se jedná o jakousi vysoká zeď, propast či podobné terénní překážky, za kterými je prostor hráčům znepřístupněn. Pro osu z je situace rozdílná. Spodní hranice je zem, takže zde je hranice jasná. Nicméně horní hranici většinou tvoří obloha. Musí se tedy stanovit maximální výška, do které objekt může vystoupat. Pro hry se členitým terénem (s prvky kopců, věží apod.) může dojít k tomu, že horní ohraničení musí být stanoveno dostatečně vysoko. To vede k tomu, že velká část prostoru bude téměř vždy prázdná. Toto ale octree skvěle řeší tím, že nějaké sekce vytvoří velmi velké (převážně

pro vzdušný prostor), zatímco prostor u země s velkou koncentrací hráčů bude rozdělen na velmi malé části. (Ericson 2004)

3.2 Podvádění

Podvádění (anglicky cheating) znamená v kontextu videoher využití softwaru k modifikaci herních pravidel ku svému prospěchu. Od jednoduchých a relativně neškodných podvodů jako je například změna barvy postavy a podobně mohou hráči se sklony k podvádění modifikovat hru, tak že jejich zbraň vyřadí nepřátele na jednu ránu, nastaví si neomezený počet životů, zvýší rychlost pohybu a celkově změní hru tak, že je téměř nemožné tyto hráče porazit, čímž se pro ostatní hráče stane hra nehratelná.

Podvádění je možné jak u offline her, tak i v online prostředí. Pokud se jedná o hry pro jednoho hráče, tak z pohledu herní komunity či vývojářů nenastává žádný problém. Veškeré modifikace hry, které podvádějící hráč učiní, učiní jen sám pro sebe. Neovlivní tím žádného jiného hráče než sám sebe. Sice ze hry bude mít jiný požitek, než jaký vývojáři dané hry měli v úmyslu, nicméně je to jeho vlastní věc. Dokonce sami vývojáři úmyslně přidávají do her možnost „podvádění“ (např. pomocí klávesových zkratek) a kódy k jednotlivým modifikacím mohou zveřejnit. Tímto způsobem může hráč snížit obtížnost hry, přeskočit úrovně, které hrát nechce, či naopak modifikovat hru, aby byla ještě náročnější. Pro online hry toto však neplatí.

Chování hráčů v online hrách ovlivňuje všechny ostatní. V případě, že nikdo nepodvádí, o vítězi hry rozhodnou schopnosti hráčů přesně tak, jak to vývojáři zamýšleli. Pokud se však vyskytne podvodník, může zkazit zážitek ze hry všem ostatním. Neporazitelní hráči s dvojnásobným množstvím životů, zrychleným pohybem či zbraní, která vyřadí hráče na jeden zásah, velmi rychle a bez obtíží vyhrají nad každým. Toto vede ke snížení popularity hry a tomu chtějí vývojáři samozřejmě zabránit.

3.2.1 Motivace podvodníků

Na první pohled se může zdát podvádění v herním prostředí jako nelogické. Jedná se o pouhou hru. Nehrají zde role peníze, zdraví či cokoliv jiného, na čem by mohlo záležet. Proč tedy mají hráči tendenci podvádět? Z výhry ve výsledku žádný benefit nemají a pokud sami

ví, že vítězství dosáhli podvodem a nejsou tedy objektivně lepší než jejich protivníci, mohlo by se zdát, že podvádění nemá sebemenší smysl. Opak je však pravdou.

Obdobně jako reálný svět, tak i život v herní komunitě má vlastní pravidla. Pokud je hráč nadprůměrně dobrý, vždy souboje vyhrává, rychle dosáhne vysokého skóre, je ostatními hráči obdivován podobně, jako kdyby ve skutečném životě dosáhl prestižní pozice v zaměstnání. Úroveň hráčů je navíc často znázorňována vzhledem postav ve hře, tudíž každý na první pohled může zjistit, že daný hráč je v dané hře velmi schopný. Paralela ke skutečnosti může být luxusní auto či drahé oblečení, kterým lidé dávají ostatním najevo úspěšný život.

A stejně jako v reálném světě se nachází lidé, kteří vysokého sociálního statusu dosáhnou podvody, tak i v herním světě se naleznou tací, kteří využijí nelegálních taktik k dosažení sociálního postavení.

Druhá motivace je pocit ze samotné výhry. Cílem hry je vyhrát. Opomineme-li jiné cíle (jako jsou vedlejší úkoly apod.), tak výhra je to jediné, o co hráčům jde. Hráči, kteří cílí na výsledek, mohou opomenout jakési morální zásady či ohled na druhé a využijí nelegální software, který jim pomůže výhry dosáhnout.

Podle moderních výzkumů bylo zjištěno, že lidský mozek nerozliší realitu od hry v tom smyslu, že pocity, které by člověk prožíval ve skutečnosti, tak prožívá i ve hře. Samozřejmě ne fyzicky (pokud je hráč zastřelen, neucítí bolest), nicméně psychika člověka zůstává nastavena stejně. Pokud tedy hráč vyhraje, zažije výlev stejných hormonů, jako když dosáhne velkého úspěchu v realitě. Naopak v případě prohry prožije porážku obdobným způsobem, jako by prohrál pěstní souboj. V průběhu hry, kdy hráč bojuje o obsazenou základnu, utíká před silnějším soupeřem, či se schovává, aby doplnil životy, prožívá obdobné emoce, jako by bránil svůj dům, utíkal před šelmou či se schovával před vrahy.

Pro mozek člověka to tedy není jen hra. Výhra je jediný cíl, kterého musí hráč za každou cenu dosáhnout. A pokud k dosažení cíle je potřeba podvod, méně uvědomělí hráči sáhnou i po tomto řešení. (Stuart 2021)

Nalézáme zde však ještě jinou motivaci a sice souboj podvodníků a vývojářů. Jak již bylo zmíněno, podvodníci snižují kvalitu hry pro ostatní hráče, kteří se mohou uchýlit k odinstalování hry. Nicméně i vývojáři mají prostředky, jak podvádění zabránit. Vzniká zde

tedy jakási nová hra mezi podvodníky a vývojáři, kdy se podvodníci snaží přechytračit a porazit vývojáře, zatímco autoři hry dělají vše pro to, aby praktiky těchto hráčů potlačili. V tomto případě není cíl podvodníků vyhrát samotnou hru, ale najít strategie, jak vývojáře přelstít – a tím vyhrát svou vlastní hru.

3.2.2 Druhy podvádění

Stejně jako ve hrách mimo digitální svět podvádět lze více způsoby. Podle článku o podvádění v online hrách (Yan a Randell 2005) lze rozdělit podvádění do více kategorií. V následujících odstavcích budou představeny jednotlivé kategorie podvádění či nežádoucího chování. Navíc budou uvedeny příklady demonstrující popsané chování a také možnosti, jak tomuto podvádění zabránit.

1. *Přístup k informacím*

Mezi klientem a serverem probíhá neustálé předávání dat o stavu hry. Program na straně klienta využívá tato data k vykreslení herní scény na obrazovku a pro ostatní algoritmy běžící u klienta. To, že zařízení hráče má přístup k datům znamená, že je podvodník může použít ve svůj prospěch. Představit si to lze na příkladu strategické hry, kdy hráči hrající na jedné mapě začínají se svým obsazeným územím. Jediné, co zatím vidí, je část jejich teritoria a zbytek je prozatím skrytý ve tmě. Aby hráči objevili další část mapy nebo viděli tahy ostatních hráčů, musí postavit pozorovatelny či posílat průzkumníky.

Počítač klienta však přijímá všechna data o hře včetně skrytých území i tazích soupeřů. Pokud podvodník tato data objeví, může rekonstruovat celou mapu s odhaleným území podle přijatých dat.

Týmž způsobem lze v případě střílecí hry určit pozice protihráče, který je schovaný za stěnou, zjistit zbraně soupeřů a jednoduše se dostat k jakýmkoliv informacím, které vývojáři zamýšleli pro hráče nedostupné.

Jeden ze způsobů, jak předejít tomuto typu podvodu, je šifrovat tyto informace nebo je neposílat vůbec. Šifrování lze potenciálně prolomit. Ještě bezpečnější je posílat jen ty informace, které jsou potřebné pro vykreslení herní scény. Naprosto nejbezpečnější by bylo posílat de facto jen výsledný obraz, který se klientům vykreslí na obrazovku bez žádných

přídavných dat. Tento přístup je však nevyhovující z hlediska rychlé responzivity (jak je popsáno v kapitole Online komunikace)

2. *Podvod dohodou*

V tomto případě se jedná o spolupráci dvou či více hráčů za účelem vzájemného profitu nelegální cestou. Jeden z příkladů této taktiky by mohla být online hra, kde hráči získají vyšší rank (tj. postavení hráče v žebříčku hry) za vítězství v bitvě. Tímto způsobem se mohou hráči dohodnout, že první hru se nechá porazit jeden hráč a následující hru zas druhý. Takto dosáhnou oba vyššího ranku, aniž by to jakkoliv odráželo jejich herní schopnosti.

Další příklad by mohla být karetní hra, kdy dva hráči se ve spolupráci snaží porazit třetího. Žádný hráč nevidí karty v ruce ostatních. Podvod nastává ve chvíli, kdy si spolupracující hráči předají informace o svých kartách, které jsou za normálních okolností skryty.

Zde toho vývojáři příliš nezmůžou. Zabránit komunikaci mezi dvěma lidmi jednoduše nelze. Samozřejmě – lze detekovat, zdali hráči nemají na svém zařízení zapnuté jiné aplikace umožňující komunikaci (sociální sítě, e-mail aj.). Avšak nelze jim zabránit, aby si zatelefonovali nebo spolu mluvili, pokud by se nacházeli ve stejné místnosti.

3. *Zneužití herních postupů*

V některých hrách lze najít různé slabiny či neošetřené případy, které hráč může využít bez žádného přídavného softwaru či jakýchkoliv znalostí o počítačové bezpečnosti. Zde by se dalo polemizovat, zdali se vůbec jedná o podvádění či pouze nemorální chování, jelikož hráč nepoužívá nic nelegálního, jen se snaží využít slabých míst ve hře či nepozornost protihráče.

Jako příklad může sloužit hra, ve které je cílem zastřelit protihráče. Z každého zastřelení hráč získává skóre. Může nastat situace, kdy si je hráč jistý, že ho soupeř každou chvíli zastřelí, tak se těsně před svou smrtí odpojí, aby z toho protihráč neměl žádné skóre. Zde se ani tolik nejedná o podvod jako spíše o pouhou zlomyslnost.

Další příklad by mohly být piškvorky online. Servery pro hraní piškvorek či podobných her často fungují tak, že klient založí místnost, kam se může kdokoliv připojit. Zakládající hráč se stává správcem, který může libovolně měnit nastavení, včetně toho, zdali se bude jednat o přátelskou hru, kdy výhra či prohra nemá vliv na rank, či o hodnocenou hru, kde výhra zvýší hráči rank, stejně jako prohra ho sníží. První hráč má v piškvorkách velkou výhodu, a

proto se po každé hře začínající hráč střídá. V momentě, kdy je začínající hráč správce místnosti, nastaví stůl jako hodnocený a jelikož začíná, má větší šanci na výhru. Těsně před zahájením druhé hry, kdy začínat bude oponent, přepíše nastavení na přátelskou hru, aby neztratil rank v případě prohry. Nezkušený hráč může přehlédnout změněné nastavení a ocitnout se v nespravedlivě horším postavení.

Zde vývojáři disponují prostředky, jak tomuto chování zabránit. V prvním případě mohou nastavit, že postava hráče bude i po odpojení ve hře a body za její zabití stále soupeř získá. V druhém příkladu by stačilo jasně varovat hráče při změně nastavení místnosti. Zde je problém spíše v uvědomění si všech slabin, které mohou nemorální hráči zneužít. Proto je pro vývojáře důležité neustále sledovat vývoj hry a získávat zpětnou vazbu od hráčů.

4. *Podvodné obchody s virtuálním zbožím*

V různých hrách je možné vydělávat virtuální měnu či různé materiály a suroviny, za které si hráči kupují herní vybavení. U jistých her je také možné předávat a směňovat zboží mezi hráči. To samotné je neškodné, dokud se nezačlení reálné peníze. Hráči, kteří dosáhli vybavení vysoké úrovně či velkého množství virtuálních peněz, mohou být ochotni prodat toto virtuální zboží za reálné peníze. Tato aktivita se nazývá Real money trading (RMT) přeloženo jako obchod s reálnými penězi. RMT je vnímáno vývojáři jako nežádoucí. Obchodníci využívají různé taktiky (účty ovládané roboty apod.), aby sehnali dostatek herního vybavení, které následně prodají. Tím pádem dojde k jakési inflaci virtuální měny ve hře, zároveň účty robotů ovládané automaticky mohou způsobit negativní interferenci s lidskými hráči (roboti například mohou sesbírat suroviny, které jsou pro hráče rozmístěné na mapě). Z těchto důvodů se vývojáři snaží potlačit obdobné chování, a to kupříkladu odstraněním herních účtů těchto lidí. Ve hře Wall to wall takedowns bylo odstraněno přes deset tisíc účtů z důvodu RMT (Labs 2021).

Z právního hlediska se také nejedná o pouhou maličkost. V Jižní Koreji dva hráči provozovali prodej herního zboží. Následně byli obviněni z nedaněného gamblerství. A ačkoliv případ nakonec vyhráli, je na celou záležitost pohlíženo negativně (AJ Glasser 2010).

Toto samo o sobě podvádění není. Podvádění nastává až tehdy, kdy podvodník dostane zaplaceno, ale slíbenou věc nedodá.

Další riziko nastává v momentě, kdy je kupující hráč vyzván, aby udal své heslo, pomocí kterého se mají dostat podvodníci na jeho účet. I v případě, kdy dohodnuté zboží prodejci dodají, kupující hráč nemá žádnou kontrolu nad tím, co s jeho heslem dále učiní.

5. *Využití umělé inteligence*

Podvodníci mohou v různých hrách použít prvek umělé inteligence. Jedná se převážně o hry, které jsou založeny na myšlení a rychlém rozhodování hráčů. Většinou se jedná o online společenské hry jako jsou například šachy, piškvorky či go. Proti kvalitní umělé inteligenci nemá lidský hráč téměř šanci.

Zde vyvstává pro vývojáře nelehký úkol rozlišit umělou inteligenci od reálného hráče. V případě šachů může být nezkušený hráč, který používá rady softwaru pro každý tah, snadno odhalen. Nicméně zkušenějšímu hráči k výhře stačí rada jen v pár zásadních situacích. Zároveň mohou podvodníci cíleně činit menší chyby, aby obvinění unikli. (Jouni Smed a Harri Hakonen 2006)

Oproti tomu by se mohlo stát, že by hranice byla nastavena příliš přísně, což by vedlo k obviňování nevinných hráčů. Přeci jen, dostatečně dobrý hráč dělá správné tahy, které se mohou shodovat i s doporučením od umělé inteligence.

Pozn.: Sám autor této práce byl vícekrát obviněn z podvádění při hraní piškvorek online, ačkoliv žádný pomocný software nepoužil. K obvinění často došlo od méně kvalitních hráčů, kteří se pod vlivem emocí z prohry snaží najít útěchu v myšlence, že je jejich prohra nespravedlivá.

6. *Podvádění modifikací klientské infrastruktury*

Hráči mohou přenastavit svůj počítač a jeho komponenty tak, aby hra byla vykreslena jiným způsobem, než vývojáři zamýšleli.

Například podvodník může přenastavit svůj grafický procesor tak, aby vykresloval zdi poloprůhledně, čímž získá oproti ostatním nefér výhodu.

7. *Přehlcení sítě*

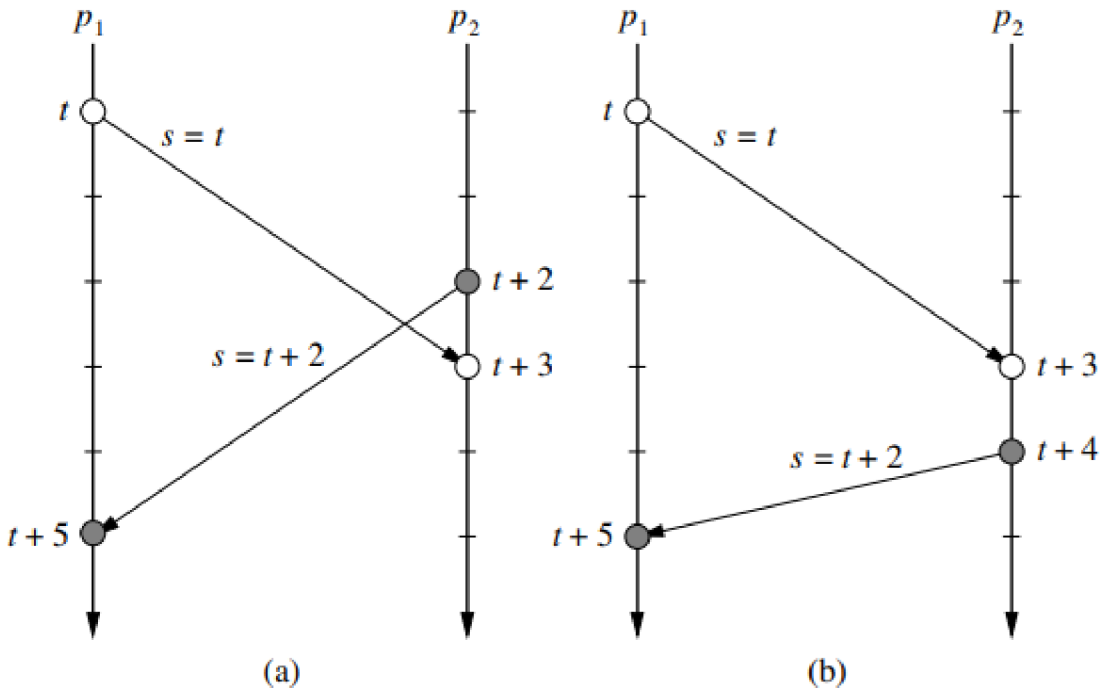
Tato strategie lze užít pouze pro hry s architekturou bez serveru (peer to peer), která bude popsána v pozdějších kapitolách. V tomto případě může podvodník zpožďovat příchozí data

od některého z hráčů (například DoS útokem). Poté se může zdát, že napadený hráč má špatné spojení, na základě čeho může být ze hry odpojen.

8. *„Look ahead" podvádění*

Při tomto typu podvádění hráč získá nelegálně informace o akcích ostatních hráčů a až teprve podle nich reaguje, ačkoliv by měl hrát synchronizovaně s ostatními. Tím pádem získává nefér výhodu v tom, že zná další tahy svých oponentů, ke kterým by za normálních okolností neměl přístup. Učiní to tak, že předstírá vyšší latenci, než ve skutečnosti má. Své akce rozhodne až ve chvíli, kdy obdržel informace od ostatních hráčů. Odsud název „look ahead" (z anglického „koukni se dopředu"). Tento typ podvádění je možný jen pro architekturu peer to peer, jelikož v opačném případě podvodník získává informace synchronizovaně od serveru a nemá tedy možnost tuto strategii využít.

Situaci lze demonstrovat na následujícím obrázku:



Obrázek 12: Look ahead

Zdroj: (Jouni Smed a Harri Hakonen 2006)

Lze uvažovat následující případ. Doba zpoždění v online komunikaci soupeře jsou 3 časové jednotky. Latence podvádějícího hráče je pouze 1 jednotka (sloupec b). Podvodník může předstírat zpoždění také 3 jednotky, čímž nebude nijak podezřelý. Na venek jeho chování

vypadá jako ve sloupci a. Při porovnání obou situací lze spatřit, že podvodník ve skutečnosti reagoval až poté, co obdržel informaci od protihráče. Zpětná reakce však dojde ve stejnou chvíli, jako kdyby měl 3x větší latenci (kterou by teoreticky mít mohl). Tímto způsobem získává nespravedlivou výhodu nad ostatními.

Jednoduchý proces, jak tomuto zabránit, je každý moment požadovat nejprve informace od všech hráčů a až teprve poté je rozeslat všem ostatním. Jiné řešení by bylo použít architekturu klient-server (Jouni Smed a Harri Hakonen 2006).

9. *Cílené využívání chyb ve hře*

Jako podvádění je často považováno i cílené zneužití chyby, která se v dané hře vyskytla. Chyba od strany vývojářů se samozřejmě objevit může. Je to však něco nechtěného, neúmyslného a nezamýšleného. A často také něco, čeho si nevšimne každý. Tudíž pokud hráči objeví takovouto nedokonalost, neměli by jí cíleně využívat. Hráč například objeví, že na jistém místě ve hře lze projít zdí a bude toho využívat k překvapivým útokům nebo jako cestu úniku. Tuto strategii ostatní použít nemohou, protože o chybě nevědí. A proto se toto chování považuje za nelegální.

Avšak hranice mezi tím, co je považováno za chybu a co za pouhou nedokonalost, je velmi tenká. Například pokud hra nabízí hráčům více zbraní, které jsou však nevybalancované, rozhodně nelze obvinit hráče z podvádění jen proto, že si vybral silnější zbraň.

Vývojáři tomuto typu podvádění mohou předejít velmi snadno – jednoduše chybu odstraní. Proto je důležité získávat zpětnou vazbu od hráčů a hru stále sledovat.

10. *Využití softwaru zvyšující hráčské schopnosti*

V tomto případě se jedná o podvádění, kdy hráč využije přídavný software, který mu pomůže během hry lépe reagovat. Typický příklad je tzv. aim bot neboli software pro automatické zaměřování nepřátel. Ve střílecích hrách tento program hráči usnadňuje míření tak, aby kurzor následoval nepřátelskou postavu. Další příklad může být trigger bot – tento software vždy automaticky vystřelí v momentě, kdy se nepřítel nachází v pozici, ze které je možné soupeře zasáhnout.

Tento typ podvádění je relativně náročný na detekci obdobně jako v případě užití umělé inteligence u společenských her. Zkušený hráč totiž také dokáže zaměřit a vystřelit ve

správný okamžik, tudíž se jen obtížně rozezná kvalitní hráč od podvodníka. Jeden ze způsobů, jak toto chování odhalit, je analyzovat akce hráče. Pokud se shodují s předpokládanými akcemi při používání tohoto softwaru, lze hráče označit za podvodníka a odstranit ho ze hry.

11. Porušení pravidel stanovených vývojáři

Různé hry mohou zveřejnit svá vlastní pravidla, podle kterých se bude hodnotit, co se považuje za nedovolené chování. Tímto způsobem vývojáři opravňují penalizovat hráče, kteří tyto pravidla nedodržují.

Jeden z příkladů může být tzv. spawn killing, neboli zabíjení při vstupu do hry. Toto chování se projevuje tak, že hráči odhadnou místo, kde se vyskytnou nově připojení soupeři a okamžitě je vyřadí ze hry, aniž by příchozí stačili reagovat. Ač toto chování přímo nespadá do definice podvádění, může být u některých her považováno za nedovolené.

Jako příklad takového kodexu může sloužit prospector code ze hry The Cycle Frontier. Zde je popsáno, jaké chování se považuje za nevhodné a jak s takovým hráčem bude naloženo. Například konkrétně v této hře je považováno za nepřípustné zvolit si vulgární jméno své postavy. Obviňování hráčů z podvádění mimo oficiální cesty je taktéž sankciováno. (YAGER Development GmbH 2023)

3.3 Online komunikace

Tématem této práce je online multiplayer hra. Z toho důvodu jeden ze základních aspektů pro úspěch musí být co neplynulejší komunikace mezi jednotlivými zařízeními. Pokud bude docházet k častým výpadkům či dlouhým prodlevám v přenosu informace, může se i ta nejlepší hra stát nehratelnou. V následujících kapitolách budou popsány různé přístupy k řešení komunikace. Zmíněny jsou i algoritmy jednoduché na implementaci, které se však kvůli nedostatečné rychlosti staly pro většinu online her nepoužitelné.

Důležitý aspekt, který zde hraje roli, je typ samotné hry. Pokud se jedná o strategickou tahovou hru, tak rychlost komunikace není fundamentální – v tomto případě vývojáři dají spíše přednost zajištění bezchybného přenosu na úkor rychlosti. Příkladem mohou být online šachy. Pokud doba mezi tím, kdy první hráč odehraje a než se tah zobrazí na druhém zařízení,

bude větší než jedna sekunda, zásadně nic se neděje a kvalita hry nebude příliš neovlivněna, přičemž latence jedna sekunda by byla u her v reálném čase neakceptovatelná. Vývojáři online her si tedy musí prvně uvědomit, jaké jsou jejich priority – zdali rychlost přenosu informace, její bezchybnost, zabezpečení proti podvádění anebo dají přednost algoritmům snadným na implementaci před pokročilými algoritmy, které jsou sice rychlejší, avšak zároveň zaberou větší úsilí a čas naprogramovat.

3.3.1 Bez serveru (peer to peer)

Přístup pro řešení online komunikace, který se používal v počátcích online her je přístup bez serveru. Při implementaci tohoto algoritmu jsou počítače klientů sobě rovny a neexistuje žádný řídící server. Pokud hráč na jednom počítači provede jakoukoliv akci, rozešle všem ostatním připojeným hráčům informaci o provedené akci. Zároveň neustále naslouchá ostatním zařízením.

Fundamentální roli v tomto přístupu hraje synchronizace hráčů. Jelikož přenos a zpracování všech informací není nekonečně rychlé, nutně dojde k tomu, že některé počítače zvládnou komunikovat a vyhodnocovat data rychleji než jiné, což by vedlo k tomu, že každý hráč uvidí hru v jiném stavu. Toto samozřejmě není žádoucí.

Výhodou tohoto algoritmu je odolnost vůči výpadkům. Pokud selže jedno zařízení, neovlivní chod hry u ostatních počítačů připojených k právě probíhající hře – jediné, co se stane, je, že hráč, který ovládal poruchové zařízení, bude ze hry odpojen. Avšak ostatní hráči mohou ve hře pokračovat dále. Oproti tomu, kdyby došlo k výpadku serveru, budou odpojeni všichni hráči.

Nevýhodou přístupu bez serveru je však potenciálně velká latence a závislost na ostatních hráčích. Například: ke hře je připojeno deset hráčů, z nichž devět vlastní bezchybný počítač se skvělými podmínkami pro bezproblémovou online komunikaci, nicméně desátý hráč je připojen přes vadný počítač se zastaralým hardwarem, u kterého je latence vyšší. Zde nastává problém. Buďto bude rychlost hry závislá na nejpomalejším článku, což způsobí to, že ačkoliv většina hráčů by mohla prožít bezvadný zážitek ze hry, jeden hráč s horšími podmínkami poškodí kvalitu hry všem ostatním. Anebo se vývojáři rozhodnou ignorovat pomalého hráče, nechat ho připojeného s tím, že jen u něj se projeví slabá kvalita připojení. Tento přístup však také není možný, jelikož se tím poruší celková synchronizace.

Jediná možnost je pomalého hráče odpojit. To však také není ideální, protože i s horšími podmínkami by teoreticky mohl hru hrát a tímto přístupem je mu to znemožněno.

Neposlední problém tohoto přístupu je již popsané podvádění – například taktika „look ahead“.

Přístup bez serveru lze tedy použít u her, kde se není důležitá rychlost vyměňovaných informací (např. tahové hry), kde se nepočítá s hráči, kteří budou podvádět (což je samozřejmě naivní přístup) a kde počet připojených hráčů k jedné hře bude relativně malý.

3.3.2 Klient–server

Architektura klient–server pracuje na rozdílném principu než výše popsaný přístup. V tomto případě zde vystupuje jeden řídící počítač – server. Server sjednocuje veškerá data přijatá od hráčů (klientů), vyhodnocuje je, synchronizuje chod hry a výsledky operací rozesílá zpět.

Velká výhoda oproti přístupu bez serveru je v první řadě synchronizace hráčů. Zjednodušeně: klienti pouze posílají informace, server je vyhodnocuje a posílá zpět. Každý klient tedy dostane stejná data jako ostatní zařízení. V případě výpadku na straně klienta server zaregistruje odpojeného hráče, avšak po navázání spojení znovu bez potíží pošle současný stav hry. Problém hráče s méně kvalitním spojením popsaný v architektuře bez serveru je snadno vyřešen. Informace posílané pomalým klientem jsou vyhodnocovány stejně jako všechny ostatní. Pouze tento pomalý hráč může pocítit větší latence v komunikaci, která je dána jeho zhoršenými podmínkami, avšak hrát stále může (i když s méně kvalitním přenosem) a ostatní hráče to nijak nepoškodí.

Problém s podvody je také snadněji řešen – server samotný kontroluje validitu přijatých dat a v případě, že obdrží data, která poslal podvádějící hráč, data zahodí a nerozešle ostatním.

Jedna z nevýhod oproti výše popsanému přístupu bez serveru je odolnost vůči výpadku. Pokud dojde k pádu zařízení na straně klienta, klient bude odpojen a hru ostatních de facto neovlivní. Pokud by však došlo k výpadku serveru, spadla by celá hra a nikdo hrát dále nemůže. Proto je běžné, že firmy herního průmyslu investují do zálohování serveru.

Samotnou architekturu klient-server lze implementovat různými způsoby, které jsou uvedeny v dalších kapitolách.

Autoritativní server

V tomto případě je veškerá moc na straně serveru. Veškeré výpočty, chod hry a jakákoliv vyhodnocení jsou prováděny jen serverem. Zařízení klientů se zde chovají jen jako prodloužené periferie – pokud hráč stiskne klávesu, serveru se pošle informace o stisku, server stisk vyhodnotí a nový stav hry pošle všem ostatním klientům, kterým se pouze aktualizuje obraz na monitoru. Autoritativní server je možné připodobnit k počítači s jednou velkou obrazovkou a mnoha klávesnicemi pro každého hráče.

Tento přístup je skvělý s ohledem na prevenci podvodů. Server přijímá jen příkazy o stisknutých klávesách. Žádné jiné informace povoleny nejsou, a tudíž zde není prostor pro podvodná data.

Na druhou stranu zde dojde ke znatelným latencím mezi stisknutou klávesou a nově přijatým stavem od serveru. Pokud doba přenosu mezi serverem a klientem je 100 ms (tento časový údaj latence mezi serverem a klientem bude pro demonstraci využíván i na dále), bude trvat 100 ms, než server obdrží informaci o vyvolané akci. Následně uběhne dalších 100 ms, než klient zaregistruje odpověď od serveru. Hráč tedy pocítí to, že stiskne šipku, po dobu 200 ms se nestane nic a až poté se postava pohne. Tomu vývojáři samozřejmě musí předejít, v opačném případě by kvalita hry značně klesla. V následující kapitole budou nastíněna řešení tohoto problému.

Tento přístup stále lze použít pro tahové hry, kdy se děj neodehrává v reálném čase (například šachy), kde není rychlost přenosu zásadní. Také by šlo uvažovat o tomto přístupu pro online hry, kdy jsou zařízení spojené na lokální síti. Latence spojená s přenosem informace bude tak minimální.

Predikce strany klienta

Při hlubším zauvažování, tahy klienta jsou téměř vždy validní a většinou tím pádem nepotřebují kontrolu ze serveru. Pokud by klient dostal na naprostém počátku současný stav hry, pro většinu akcí by vůbec nepotřeboval dostávat žádné jiné informace. Pokud stiskne klávesu vpravo, je velmi pravděpodobné, že doprava jít může. Pravděpodobnost, že by se zrovna v ten moment vedle něj objevil jiný hráč, je minimální.

Pokud je hra dostatečně deterministická, lze uvažovat, že akce provedené klientem budou korektní. V rámci uvažování tohoto ideálního modelu lze vše, co klient vykoná, okamžitě

provést bez potřeby zpětné kontroly od serveru. Pokud tedy klient stiskne klávesu vpravo, informaci pošle serveru a posune se doprava, aniž by se nechal serverem zkontrolovat. Při uvažování latence 100 ms (což je stále optimistický přístup) klient předvídá budoucnost 100 ms dopředu a ve většině případů bude mít pravdu. Tento přístup tedy předpokládá, že program hry, který běží na serveru, bude běžet zároveň i u klienta. Pokud například klient narazí do objektu, může vyhodnotit kolizi, aniž by potřeboval kontrolu ze serveru.

Avšak stále nastanou případy, kdy klient budoucnost odhadne nesprávně. Může se jednat například o situaci, kdy hráč běží dopředu, ale soupeř skočí přímo před něj. Klient tedy předpokládal, že před ním bylo stále místo, ačkoliv ve skutečnosti se tam objevil jiný hráč. Ve výsledku by se oba hráči objevili na sobě.

Tomu lze předejít neustálou kontrolou se serverem. Pokud klient provede akci, ihned se stav aktualizuje, ale zároveň se přepošle zpráva serveru. Server (který v daný moment zná reálný stav hry) vyhodnotí, zdali byla akce validní a pokud ne, klienta opraví. To může vést k drobným grafickým nedokonalostem, kdy je nejprve ukázán nesprávný obraz, který je rázem opraven.

Problém nastává v momentě, kdy je třeba synchronizovat data více klientů. Lze si představit situaci, kdy je latence mezi klienty a serverem 100 ms. První hráč se pohne dopředu – pohyb je validní. Za 100 ms přijde zpráva serveru, že hráč je na nové pozici. Server tomuto klientovi potvrdí novou pozici a zároveň všem ostatním přepošle pohyb hráče. Mezi momenty, kdy první hráč stiskl klávesu a druhý hráč jeho pohyb zaznamenal, uběhlo 200 ms. Hráči sebe navzájem tedy vidí v jiný časový okamžik, což narušuje základní požadavek na jakoukoliv online hru – synchronizace.

Potvrzení informací

Na úvod této podkapitoly budou shrnuty dosavadní návrhy. Latence způsobená přenosem dat je neměnná. Čekat 100 ms (či více) na potvrzení pohybu ze strany serveru je neakceptovatelné. Nabízí se tedy přístup popsaný v předchozí podkapitole, avšak u něj je fatální nesynchronizovaná data – je to však nejpřijatelnější alternativa, ačkoliv zdaleka ne ideální či 100 % spravedlivá vůči všem hráčům.

Synchronizace dat probíhat stále může (a de facto musí) alespoň na straně serveru. Server vyhodnocuje a kontroluje všechny akce provedené hráči. Klient zobrazuje své akce ještě

předtím, než obdrží potvrzení ze serveru a pokud cokoliv zobrazí klient nesprávně, za 100 ms vyvolanou akci napraví. Z tohoto důvodu je vhodné zásadní a z hráčského pohledu nevratné akce nevyhodnocovat ihned, ale počkat na zprávu od serveru.

Znovu pro připomenutí – hráči se navzájem vidí s 200 ms zpožděním a pouze server vidí vše absolutně správně.

Jako příklad může sloužit souboj dvou hráčů. První hráč prohrává a má poslední zbytek života. Nepřítel vystřelí a hráče zasáhne. Ten však pár milisekund před fatálním zásahem stihne aktivovat štít, který ho před střelou ochrání. Nepřítel však o štítu ve svém světě neví. Z jeho pohledu zasáhl, objeví se animace umírajícího hráče, přičtou se mu body za eliminaci hráče, aktivují se mu nové schopnosti apod. Nicméně server po srovnání příchozích dat od obou hráčů vyhodnotil, že napadený hráč opravdu stihl aktivovat štít včas a stále žije. Pro napadeného hráče je vše v pořádku. V jeho světě je před střelou uchráněn, stále žije a server mu to jen potvrdil. Druhý hráč však až nyní obdržel informaci, že hráč není mrtvý. Jeho získané body se zas odečtou, přijde o právě získané schopnosti a z ničeho nic mrtvý hráč, kterého pár milisekund nazpět zastřelil, znovu vstává a běhá, jako by se nic nestalo. Lze předpokládat, že podobné situace nejsou žádoucí. U důležitých momentů jako je právě zmíněné vyřazení hráče nebo třeba obsazení základny či dosažení cílové pásky je vhodné počkat na potvrzení od serveru.

Kompenzace latence

Je stále dobré mít na paměti, že hráč vykoná akci, 100 ms zpráva putuje k serveru, který vyšle informace, které ostatním hráčům cestují dalších 100 ms. Z důvodu synchronizace je server určen jako ten, u kterého probíhá kontrola všech dat, takže lze říci, že na straně serveru probíhá hra „v přítomnosti“. Klient sám sebe tedy vidí 100 ms „v budoucnosti“, zatímco všechny hráče 100 ms „v minulosti“. Pro většinu situací je 200 ms stále ještě dostatečně krátký čas na to, aby to způsobilo větší potíže. Avšak pro časově a prostorově senzitivní situace mohou i detaily hrát roli. Znovu je uveden příklad souboje dvou střelců. Jeden hráč běží na otevřeném prostoru směrem k úkrytu. Druhý hráč na něj míří puškou s přesným hledím a snaží se o zásah hlavy. Nepřítel je zaměřen, hráč vystřelí. Ve světě tohoto hráče šlo o výstřel, který nemohl minout, avšak i přesto server vyhodnotí, že k zásahu nedošlo, ačkoliv odstřelovač mířil přesně. V momentě výstřelu byl totiž běžící hráč viděn 100 ms v minulosti. Navíc jelikož hráč sám sebe vidí v budoucnosti, i výstřel byl serverem zaznamenán až 100

ms po stisku klávesy. Za 200 ms se napadnutý hráč už stihl schovat do úkrytu, aniž by byl jakkoliv zraněn.

Takovýto přístup by byl pro spoustu her neakceptovatelný. Z přesného míření by se stala loterie, zdali náhodou střela protivníka zasáhne či nikoliv. Vývojáři online her však i tento problém vyřešili. S každou akcí bude serveru poslán i přesný čas jejího vykonání. Server tedy může vyhodnotit, zdali výstřel měl nepřítele zasáhnout nebo minout. A vyhodnocovat to logicky musí z pohledu hráče, který právě střílí, jinak by střílejícímu hráči bylo přesné míření stále k ničemu. Co se ve skutečnosti tedy stane, je to, že střílející hráč vidí hráče běžet na volném prostoru, zaměří a vystřelí. Běžící hráč je ve svém světě již o kousek dál, avšak je přesto zasažen, protože z pohledu střelce by zastřelen být měl.

Zde dochází k jedné zásadní nepřesnosti v neprospěch zasaženého hráče. Hráč běží, tuší přítomnost odstřelovače, a proto se co nejrychleji snaží schovat do skrýše. Nakonec se mu to podaří, je v bezpečí za zdí, kde ho nikdo nemůže zasáhnout. Avšak za pár milisekund, kdy je už schovaný za skrýší, dostane zásah do hlavy, aniž by chápal, jak je to možné.

Znovu nastal časový problém – sám sebe vidí o 100 ms před serverem, takže ačkoliv je ve svém světě už ve skrýši, server ho vidí teprve vstupovat do skrýše a střílející hráč ho ale stále spatřuje běžícího na otevřeném prostoru. Jelikož je zásah vyhodnocován z pohledu střelce, je hráč schovaný ve skrýši zasažen v momentě, kdy pro střelce stále běžel ještě na otevřeném prostoru i přesto, že si on sám myslí, že je v bezpečí.

Tento přístup stále není stoprocentně spravedlivý pro obě strany, avšak je to pravděpodobně ten nejlepší kompromis mezi všemi dříve zmíněnými přístupy. (Gambetta 2022)

3.3.3 Přenos informace

Ať už je zvolen jakýkoliv přístup, je třeba se zamyslet nad tím, jaké informace budou přenášeny, a hlavně jakým způsobem. V dnešní době existuje více technologií a přístupů, jak realizovat přenos dat.

Formát dat

Základní pilíř, na kterém stojí komunikace mezi více zařízeními, je shodný formát dat. Veškerá data jsou posílána pouze jako sled bitů. Odesílatel i příjemce tedy musí mít nastavená stejná pravidla, aby přijatá zpráva mohla být dekódována.

Nejprimitivnější řešení je vše nastavit ručně. Například pokud se mají přeposlat data o určitém hráči, lze nastavit, že první 2 byty budou představovat počet jeho životů, následujících 8 bytů jeho jméno a takto by se dalo pokračovat, dokud nebude přeposláno vše.

Na první pohled je jasné, že tento přístup je nevyhovující. Vývojář by strávil spoustu času nastavováním formátu všech dat.

Efektivnější přístup je použít existující technologie, které tento problém řeší. Naskýtají se dva způsoby, jak data posílat. Buďto v textovém formátu, kde jsou data čitelná i člověkem, anebo binárně, kde je poslán sled bitů, které dohromady tvoří celou zprávu. Tento formát není čitelný člověkem, pouze programem. Pozitiva a negativa obou přístupů budou popsány u příkladů těchto technologií.

Jeden z běžně využívaných formátů (který je mimo jiné využit i v demonstrativní hře) je JSON formát.

JavaScript Object Notation (dále JSON) je způsob kódování informace po vzoru objektů z programovacího jazyka JavaScript. Veškerá data jsou uložena ve formátu klíč-hodnota. Například pokud by se kódovali informace o hráči se jménem „Player1“ se třemi životy a s dosaženým skóre 2000, zpráva v JSON formátu by mohla vypadat následovně:

```
{
        "name": "Player1",
        "lives": 3,
        "score": 2000
}
```

Tento formát je vhodný pro objektově orientované jazyky. Ačkoliv vychází z jazyka JavaScript, lze ho použít i u jiných jazyků jako je například C#. Data o objektech lze snadno převést do JSON formátu, poslat a na straně příjemce převést zas zpět. (MDN Contributors 2023)

Výhoda a zároveň nevýhoda tohoto formátu je ta, že vše je přenášeno pomocí čitelného textového řetězce. Při ladění programu, kontrolování či při hledání chyb může být snadná čitelnost výhodou, kdy programátor na první pohled vidí, jaká data jsou přenášena a kde se mohla potenciální chyba vyskytnout.

Na druhou stranu se zde vyskytuje problém se serializací a deserializací dat (neboli převodem dat do JSON formátu a z JSON formátu zas zpět). Dekódování informací z textového řetězce je výpočetně náročnější než čtení binárních dat. Oproti prvnímu návrhu, kdy jsou data posílána podle vlastních pravidel a přímo čtena bez nutnosti je jakkoliv převádět, je v případě použití JSON formátu nejprve potřeba zjistit, o jaká data se jedná (např. životy hráče), přečíst hodnotu (počet životů je 3) a přiřadit ji (daný hráč má 3 zbývající životy). Tento proces zabere pochopitelně určitou dobu. Pro menší množství dat je však zpoždění způsobené deserializací dat zanedbatelné. Druhé negativum posílání pomocí čitelného textu je bezpečnost. Pokud jakýkoliv připojený klient může přečíst vše, co je právě přeposíláno, může tyto informace zneužít ve svůj prospěch – jakékoliv podvody jsou mu tímto značně usnadněny.

Další alternativa je použití MessagePack. Tato technologie je obdobná JSON formátování. Principiálně funguje stejně – vstupní data převede do svého formátu, informace přepošle a na straně příjemce zas data dekóduje. Rozdíl je zde ve velikosti zformátovaných dat. MessagePack nepoužívá k formátování přístup klíč-hodnota. Data ukládá ve své původní podobě do jednoho řetězce bitů. Tím pádem je délka jednotlivých zpráv menší než v případě JSON formátu. Zároveň i doba serializace a deserializace je kratší vzhledem k tomu, že data není potřeba převádět z textové podoby.

Při testech porovnání MessagePack a JSON vyhrává MessagePack. Deserializaci testovacích dat trvala při použití JSON přes 30 sekund. Stejnou operaci MessagePack stihl za pouhých 7,3 sekundy. Obdobně je to s pamětí. Data zpracovaná JSON formátem měly velikost 143 025 bytů. Stejná data MessagePack zpracoval na velikost 120799 bytů, což je přibližně o 15 % lepší výsledek. (Nickolas Da Silva 2020)

Jak již bylo zmíněno, MessagePack nepoužívá textový formát, nýbrž binární. V praxi to znamená, že posílané zprávy nejsou pro člověka čitelné. To může do jisté míry ztížit ladění programu vývojářem, avšak přeposílaná data jsou primárně určena ke komunikaci mezi zařízeními, nikoliv lidmi. To je primárně výhodné z hlediska zabezpečení před uživateli se sklony k podvádění. Binární řetězec lze dále zašifrovat, čímž se zpráva samotná stane pro podvodné účely nevyužitelná.

Mezi další alternativy, které je vhodné zmínit, patří technologie YAML, Avro, OData či Protobuf.

Krok, který dále pomůže zredukovat velikost přeposílaných zpráv, je užití kompresního programu. Existuje řada kompresních algoritmů a programů, které se dají využít k zmenšení velikosti posílaných dat. Na příklad lze uvést program Gzip, který užívá kompresní algoritmus LZ77, který typicky redukuje vstupní řetězec o 60 % až 70 %. Samozřejmě i zde snižuje použití kompresního programu celkovou rychlost programu. Komprese a následná dekomprese zabere jistou dobu, avšak z praktického hlediska je tento čas zanedbatelný (Jean-loup Gailly 2022)

Algoritmy pro přenos informace

Jeden z nejnaivnějších přístupů (a současně z nejpomalejších), je posílat všechna data všem hráčům. Klienti serveru posílají informace o svých akcích (pohyb apod.), server zas všem klientům přeposílá kompletně všechna data ze hry.

Výhoda tohoto algoritmu je snadná implementace, celistvost informací a resistence vůči výpadku – pokud dojde ke krátkodobému přerušení komunikace, klient hned po opětovném připojení znovu obdrží kompletní stav hry.

Tento algoritmus je však neakceptovatelně pomalý. Každý hráč má spousty vlastností (životy, velikost postavy, skóre, barva postavy, dosažená vylepšení apod.), u kterých můžeme předpokládat, že se příliš často nemění. V tom případě se neustále posílají redundantní data. Pokud informace zůstávají po většinu času stejná, stačí posílat jen změny stavu. Každý snímek tedy server nemusí posílat, že hráč A má stále 10 životů, když je měl i poslední dvě minuty. Teprve až o jeden život přijde, server pošle ostatním informaci o tom, že má životů jen devět.

Zde byl již naznačen druhý algoritmus – posílání změn. Místo toho, aby se posílala samotná data, budou se posílat téměř výhradně změny. Po připojení klienta do hry pošle server informaci o celkovém stavu hry. Nadále server bude posílat jen změny. Program hry, který běží na serveru, může běžet i u klienta. V tom případě klient dokáže zpracovat všechny informace, které mu server pošle, a tudíž už nemusí posílat vůbec žádná data, pouze změny.

Zároveň se zprávou o změnách ve hře se pošle přesný čas, kdy tyto změny proběhly. Tím pádem může klient přesně vyhodnotit stav hry bez pomoci serveru. V tomto případě se server stává jen jako jakýsi prostředek komunikace. Server má však samozřejmě hlavní slovo a

pokud by došlo k jakýmkoliv neshodám mezi klientem a serverem, tak pravda je na straně serveru.

Zde však vyvstává slabina algoritmu, která se u přístupu „poslat vše všem“ nevyskytla. V případě, že dojde ke krátkodobému přerušení spojení, dojde zároveň ke ztrátě informací o změně. Pokud klient neobdrží zprávu o jedné ze změn, budou všechny následující zprávy vykonány nesprávně. Například hráč A jde dopředu a následně zahne doleva na úzký most. Hráč B neobdrží informaci, že šel dopředu. Pouze dostane zprávu, že zahnul doleva. Jeho pozice je však nesprávná a hráč B by vyhodnotil, že hráč A spadnul z mostu.

Tomu lze předejít tím, že ke každé zprávě bude poslán index zprávy. Index se po každé poslané zprávě zvýší o jedna. Pokud tedy klient obdrží zprávu, jejíž index nenásleduje přesně po posledně přijaté zprávě, požádá server o zopakování poslání ztracené zprávy. Pokud dojde k delšímu výpadku, může klient požádat o poslání všech dat o hře (což může být rychlejší než rekonstruovat všechny kroky, které se staly během přerušení).

Technologie pro bezdrátový přenos dat

Bezdrátová komunikace mezi jednotlivými zařízeními probíhá vždy na stejném základě – z jednoho zařízení je odeslán sled bitů, kde každý bit je zakódován pomocí vlastností elektromagnetického vlnění (frekvence, amplituda, fáze). Na druhém konci příjemce toto vlnění zaregistruje, převede do binárního kódu a ten nadále může zpracovat.

Nyní vyvstává otázka, jak konkrétně tento přístup implementovat. Je nutné podotknout, že pravděpodobnost výskytu chyby u bezdrátové komunikace je značně vyšší než při drátovém spojení (PathFinder Digital LLC 2020). Z tohoto důvodu je v různých situacích vhodné data kontrolovat ať už posláním kontrolních bitů či zpětnou kontrolou od serveru. To však zvyšuje velikost odesílané zprávy o redundantní bity a zároveň zvyšuje čas celkového zpracování informace čekáním na odezvu ze serveru.

Pro některé účely může být vhodnější smířit se s občasným výpadkem dat za cenu vyšší rychlosti.

Základní dva užívané protokoly dnešní doby jsou TCP a UDP. TCP užívá prvně zmíněný přístup – data neustále ověřuje, pokud nalezne chybu, nechá si zprávu poslat znovu. Oproti tomu UDP dává přednost rychlosti za cenu možné ztráty dat, což se například hojně využívá při přehrávání online videí – cíl je vysoká rychlost. Pokud náhodou klient neobdrží jeden

snímek, kvalitu to příliš nezhorší (jako by zhoršilo dlouhé čekání na to, než se načte další obsah)

K ještě snadnější implementaci online komunikace jsou vytvořeny knihovny a další pomocné technologie, které vývojářům nabízejí jednoduché prostředky, jak online spojení spravovat.

Mezi technologie, které je jistě vhodné zmínit, patří protokol http (hypertext transfer protokol). Jedná se o protokol používaný pro komunikaci na internetu. Dnes je již častější setkat se s bezpečnější verzí https (secure http, který komunikaci šifruje). Tento protokol určuje formát, jakým jsou data posílána mezi klientem a serverem. V momentě, kdy klient navštíví internetovou stránku, přepošle serveru požadavek - tzv. HttpRequest. Server požadavek zaznamená a přepošle zpět odpověď, ve které je obsažený kód webové stránky společně se všemi dalšími daty jako jsou obrázky, hudba apod. Spojení serveru a klientu při užití defaultního http po přeposlané zprávě končí a pokud klient bude znovu požadovat nové informace od serveru, musí navázat nové spojení od začátku.

Tento přístup může být vhodný pro prohlížení webových stránek, nicméně pro online hru by bylo vhodnější nechat spojení otevřené – přeci jen, data mezi serverem a klientem jsou vyměňována obvykle šedesátkrát za sekundu. Takovýto přístup je možný implementovat i pomocí http metodou „Keep-alive“, kdy zůstane spojení navázané nebo například knihovna SignalR, která je mimo jiné užita v demonstrativní hře. Hlavní rozdíl oproti defaultnímu http je v neukončení spojení mezi klientem a serverem po dobu komunikace, což je pro programování online hry užitečné.

3.4 Nástroje pro tvorbu online her

Videoherní průmysl se od svých počátků v polovině 20. století značně rozvinul. S přibývajícím počtem herních vývojářů se objevila potřeba vytvoření nového softwaru uzpůsobeného k tvorbě samotných počítačových her. V momentě, kdy herní vývojáři nemusí psát kód hry od nuly, ale mají připravené prostředí ke své tvorbě, mohou se plně věnovat vývoji samotné hry. Nemusí stále dokola programovat detekci kolize – vždyť ta už byla naprogramovaná mockrát, proč by tedy nemohli použít hotový kód? Nemusí řešit formátování dat – stačí použít existující technologii, například JSON. S aplikacemi pro

tvorbu her se programování zpřístupnilo začátečníkům. A dokonce dnes již existují programy pro tvorbu her, které slouží primárně k výukovým účelům, aby se žáci seznámili se světem programování skrze něco tak populárního, jako jsou počítačové hry.

Naprostým základem pro tvorbu jakéhokoliv programu (včetně počítačových her) je programovací jazyk, pomocí kterého zadává vývojář instrukce procesoru, který krok po kroku program vykoná. Pokročilejším nástrojem, který zas o něco zjednodušuje vývoj her, je tzv. framework. Framework si lze představit jako předpřipravený základ pro vývoj softwaru. Programátor se tedy nemusí zabývat vedlejšími problémy, ale plně se soustředit na tvorbu své aplikace. Například pro vývoj desktopových aplikací nemusí vývojář řešit, jakým způsobem je detekován stisk klávesy nebo na jakém principu program pozná kliknutí myši. Programátor jednoduše napíše příkaz, který mu vrátí polohu kurzoru a nemusí od nuly programovat vše sám.

Framework je typicky spojen s programovacím jazykem.

O úroveň dále stojí herní engine (někdy též jako „herní motor“, rozšířenější název i v české literatuře je však „herní engine“). Herní engine je software přímo určený pro tvorbu videoher. Obdobně jako použitím frameworku se nemusel vývojář zdržovat programováním opakujících se problémů, které autor frameworku vyřešil za něj, tak užitím enginu vývojářovi odpadá nutnost programovat základní algoritmy videoher – herní engine mu nabízí přímo použít předpřipravené kódy.

V následujících třech kapitolách budou popsány právě zmíněné technologie, bez kterých by se vývoj videoher v dnešním světě neobešel.

3.4.1 Programovací jazyk

Veškeré programy běží na procesoru. Velmi zjednodušeně – procesor přijme kód programu v podobě sekvence jedniček a nul, podle této sekvence vykoná odpovídající příkazy a výsledky těchto operací odešle dalším zařízením, například monitoru, na kterém se uživateli zobrazí nový snímek jeho hry.

Tato sekvence jedniček a nul se nazývá strojový kód. Tento kód není čitelný člověkem a zároveň není přenositelný mezi zařízeními – každý typ procesoru funguje jinak a strojový kód pro jeden procesor nemusí být kompatibilní s druhým – i pokud se jedná o stejný

procesor, tak hardware jednoho počítače nemusí být shodný s druhým. Psát kód programu v sekvenci jedniček a nul by v první řadě nebylo přenositelné mezi jednotlivými zařízeními a v druhé řadě prakticky velmi náročné.

Z tohoto důvodu byl vyvinut jazyk symbolických adres (assembly language). Tento jazyk používá příkazy čitelné člověkem – slova či zkratky vycházející z angličtiny (např. mov, end, jmp) – které jsou následně převedeny do strojového kódu pomocí tzv. asembleru. Tomuto kódu již procesor rozumí. Bohužel ani programy tohoto jazyka nejsou přenositelné mezi zařízeními.

Krok kupředu byl nastal při představení macro instrukcí – jednalo se o skupiny příkazů, které byly následně rozděleny do jednotlivých instrukcí. Například často opakovaný příkaz pro součet SUM A, B, C sečetl čísla A a B a uložil je do C. To nahradilo příkazy:

LOAD A

ADD B

STORE C

Stále se však jedná o malý krok dopředu. Velký pokrok v oblasti digitálních technologií znamenal příchod vyšších programovacích jazyků. Tyto jazyky užívají příkazy vycházející z angličtiny (obdobně jako u jazyka symbolických adres), k tomu však používají běžně užívané matematické symboly (+, -, *, /), čímž jsou se staly ještě srozumitelnějšími pro programátory a jednoduššími na naučení. Tyto jazyky jsou navíc již z velké části nezávislé na konkrétním zařízení. Stačí pouze jeden kód, který lze spustit na různých zařízeních bez ohledu na typu procesoru či jiného hardwaru.

Portabilita kódu je zajištěna tak zvaným překladačem (compiler), který zná instrukční sadu konkrétního procesoru a tím pádem dovede přenést příkazy z vyššího programovacího jazyka do strojového kódu spustitelného procesorem. Celý program je nejprve celý přeložen do strojového kódu a následně spuštěn – tento proces se nazývá sestavení (build).

Obdobný přístup využívá tzv. interpret, který obdobně jako překladač slouží k převedení srozumitelných příkazů vyššího jazyka do jazyka procesoru. Na rozdíl od překladače však nepřevede celý kód zaráz, nýbrž vždy jen přeloží právě očekávaný příkaz. Jakmile se splní, přeloží zas další řádek kódu. Výhoda oproti kompilovanému jazyku je primárně v tom, že program nemusí být přeložen celý při sebemenší změně v kódu, jako tomu je u

kompilovaných jazyků, kde je kód pokaždé překládán celý, což je samozřejmě časově náročnější. Na druhou stranu interpretované jazyky jsou obecně pomalejší vzhledem k tomu, že každý příkaz je překládán v průběhu programu, a ne předem.

Mezi kompilované jazyky patří například C# nebo C++. Do skupiny interpretovaných jazyků patří JavaScript či Python (Harvey M. Deitel a Barbara Deitel 1986).

Programovací jazyky lze dále dělit na staticky či dynamicky typované. Rozdíl mezi těmito skupinami hrál velkou roli při volbě konkrétního programovacího jazyka pro tvorbu demonstrativní hry, kde byl nakonec zvolen staticky typovaný jazyk C#.

V programovacích jazycích jsou hodnoty ukládány do tzv. proměnných (variables). Proměnné mají typ (například číslo nebo text) a hodnotu (například číslo 10 nebo řetězec „Ahoj“).

U dynamicky typovaných jazyků se typ pojí k hodnotě, a ne k proměnné samotné. Proměnná v dynamicky typovaných jazycích (jako je například JavaScript) je vytvořena tímto způsobem:

```
variable = 5;
```

Tímto způsobem je vytvořena proměnná se jménem „variable“ a je k ní přiřazena hodnota 5. V tento moment je proměnná typu „číslo“. Mohu k ní přičíst jiné číslo či ji použít k ukládání jiných čísel. Dynamicky typovaný jazyk však programátorovi umožňuje měnit typ proměnné. Tudíž není žádný problém napsat toto:

```
variable = „Ahoj“;
```

Tím se právě změnil typ této proměnné z čísla na text. A s textem lze provádět zas jiné operace, jako například zjistit první písmeno a podobně. Na jednu stranu tento přístup disponuje větší svobodou, avšak tím, že není zaručen typ proměnné, může se stát, že program se bude snažit přičíst ke slovu „Ahoj“ jedničku nebo zas u čísla 5 zjistit druhé písmeno. Pokud dojde k podobnému případu, většinou bude program zastaven z důvodu chyby, nebo se může snažit pokračovat dál a „popasovat se“ s nelogickými kombinacemi. Například příkaz

```
5 + „ahoj“
```

by v jazyce JavaScript vrátil hodnotu „5ahoj“ typu text.

Dynamicky typované jazyky jsou často oblíbené mezi začátečníky, kteří jsou v začátcích ušetřeni celkové problematice typů proměnných.

Na druhé straně staticky typované jazyky přiřazují typ k proměnným rovnou při jejich tvorbě. Ve staticky typovaném jazyce jsou proměnné tvořeny tímto způsobem:

```
int number = 5;
```

Tímto řádkem kódu byla právě vytvořena proměnná se jménem „number“ typu int (celé číslo) a byla jí přirazena hodnota 5. Tím je přirazen typ k této proměnné a po dobu existence tohoto programu bude proměnná „number“ vždy číslo a nikdy ne text nebo objekt nebo desetinné číslo. K tomu se pojí i následné využití této proměnné. Například ji lze sečíst s jiným číslem, ale nelze z ní přečíst druhé písmeno (protože to není text), jako to bylo možné u dynamického přístupu.

Z hlediska vývoje videoher se autor spíše kloní ke staticky typovaným jazykům, které přináší pevný řád a jasnou strukturu. V online počítačové hře je třeba mnoho funkcí, které na sebe jakýmsi způsobem navazují. Staticky typovaný jazyk neumožní vývojáři vytvořit zbytečnou chybu tím, že do funkce, která potřebuje dvě čísla, by vložil číslo a text. (Zhou 2018)

3.4.2 Framework

O úroveň výše stojí framework. Jedná se o strukturu, která pomáhá vývojáři programovat stabilní kód bez nutnosti řešit různé problémy, které za něj framework již vyřešil. Framework také vyžaduje specifickou strukturu projektu, kterou vývojář musí dodržovat. Framework by mohl být připodobněn ke kostře, na kterou vývojář připojuje svaly a maso, čímž vytvoří živočicha. Pokud má ale kostra 4 končetiny, špatně z ní vyrobí stonožku.

Lze zmínit webové frameworky (například Express), které nabízí programátorovi jednoduché příkazy k navázání online komunikace. Existují také frameworky pro tvorbu mobilních aplikací, například Xamarin – zde programátor vytváří vzhled stránky pomocí jazyka XAML, zatímco programuje v jazyce C#. Tyto dva koncepty propojuje přesně danými pravidly. Na druhou stranu však nemusí řešit, jak je kód nahrán do mobilního zařízení nebo jakým způsobem reagují komponenty na akce od uživatele – jednoduše napíše příkaz a zbytek je již vyřešen.

K frameworku se pojí programovací jazyk (či jazyky). Zvolený webový framework pro tvorbu demonstrativní hry Blazor ku příkladu využívá jazyk C# a okrajově JavaScript.

Pojem framework by neměl být zaměňován s knihovnou. Jako knihovna je v programování nazýván externí kód obsahují obecnou funkcionalitu, který následně programátor může využít pro tvorbu konkrétní aplikace. Například existuje knihovna pro vytvoření videa ze sekvence obrazů. Tato knihovna tedy není spustitelný program, pouze sada funkcí, kterou vývojář může užít pro práci s videem.

3.4.3 Herní enginy

Ještě o krok dále nad frameworkem stojí herní engine – software určený přímo pro tvorbu videoher. Herní engine disponuje předpřipravenými funkcemi pokrývající běžné herní algoritmy jako je pohyb, detekce kolizí, správa objektů ve hře, tvorba úrovní, nahrávání textur a podobně. Některé herní enginy dokonce nepoužívají žádný konkrétní jazyk – k programování nabízí grafické prostředí, kde vývojář pouze vybírá a přetahuje bloky s funkcemi, řadí je za sebe a tím programuje, aniž by napsal jediný řádek kódu. Tímto způsobem může být programování představeno i malým dětem, které pomocí intuitivních obrázku vytvoří vlastní jednoduchou hru.

Herní enginy obdobně šetří čas programátorům, kteří se nemusí zabývat množstvím již naprogramovaných algoritmů. Tím, že herní engine drží pevnou strukturu, jak v něm má vývojář pracovat, se předejde mnoha chybám, programátor je intuitivně veden a jednoduše tvoří svou hru. S tím však plynou i nevýhody. Pevná struktura enginu zas do jisté míry zabraňuje programátorovi přidat svou vlastní funkcionalitu či přeprogramovat již hotové algoritmy přesně podle svých potřeb (což je jeden z důvodů, proč pro tvorbu demonstrativní hry nebyl použitý herní engine).

Příklady herních enginů jsou Unity, Unreal, GameMaker či engine určený primárně pro děti Scratch.

4 Návrh hry

4.1 Game design demonstrativní hry

Pro demonstraci výše zmíněných algoritmů byla vytvořena hra nesoucí jméno Fanior. Jedná se o akční online hru v reálném čase pro více hráčů. Hru je možná spustit bez instalace přímo v prohlížeči. Po spuštění hry se hráč připojí do arény s ostatními protihráči. Cílem hráče je přežít co nejdéle a získat co nejvyšší skóre. Hru žádným způsobem nelze přímo vyhrát.

Grafika hry je dvoudimensionální, pohled z ptačí perspektivy.

Atmosféra hry je laděna do fantasy tématiky. Hráči si mohou vybírat středověké zbraně (prak apod.) či magii (mrazící kouzlo apod.). Samotný název Fanior zní (nebo alespoň má znít) magicky.

4.1.1 Popis hry

Hráč se se svou postavou může pohybovat po 2D herním poli a využívat zbraň pro souboj s protihráči. Na začátku hry hráč začíná se základní zbraní. Po získání určitého skóre hráč může vylepšit svou zbraň, získat novou schopnost, zesílit útok, zrychlit pohyb, zvýšit maximální počet životů, zrychlit regeneraci apod. Zbraně jsou převážně střelné.

Pokud je hráč zasažen, sníží se počet jeho životů (v závislosti na vylepšení protivníkovi střely). V případě, že hráč přišel o všechny své životy, právě rozehraná hra pro něj končí, nicméně může se ihned připojit do téže arény a začít hrát od znova. Pokud je hráč zasažen, ale stále přežil, jeho životy se časem obnoví.

Ve hře se také vyskytují počítačoví nepřátelé. Ti ve hře mají roli dalších oponentů, útočí na lidské hráče a stejně jako ostatní hráči mohou být zničeni.

Hlavním herním prvkem je skóre. Skóre slouží jako jediný cíl, kterého chtějí hráči dosáhnout – skončit s co nejvyšším počtem bodů. Hráč zvýší své skóre za zabití jakéhokoliv nepřítele jak lidského, tak počítačového, přičemž čím bylo vyšší skóre zabitého protivníka, tím vyšší skóre hráč získává.

Další způsob získání skóre (jistě jednodušší, avšak pomalejší) je sbírání náhodně rozmístěných drahokamů nebo rozbíjení různých objektů svými střelami. Tímto způsobem mají nově připojení hráči možnost se vylepšit, protože je jen malá šance, že by dokázali porazit silnější hráče.

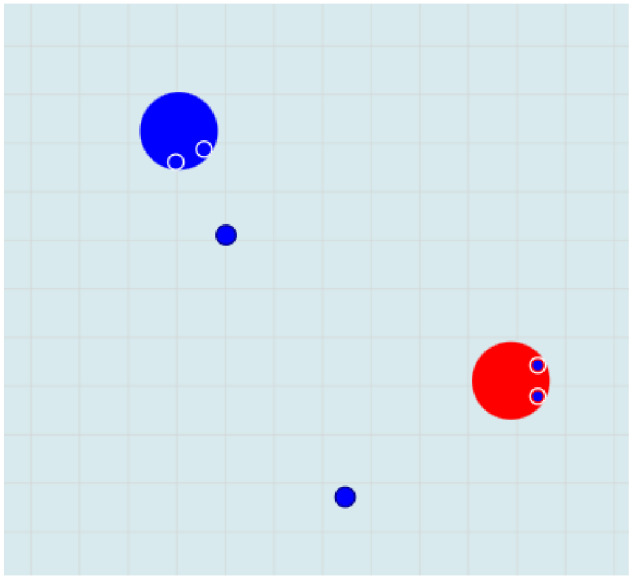
Po dosažení určitého skóre je hráči nabídnuta možnost vylepšení. Hráč se může rozhodnout vybrat si jedno z nabídnutých vylepšení (například zvýšit svůj počet životů) nebo nabídku ignorovat a šetřit si na cennější bonus. Větším bonusem se myslí vylepšení své zbraně nebo získání schopnosti. Schopnosti může mít hráč celkem 2. Jedná se o doplňující zesílení hráče na krátký čas (například dočasné zvýšení rychlosti). Zbraně se s každým vylepšením větví do dalších kategorií. Pokud si například hráč vylepší svou zbraň na luk, jako další vylepšení se mu nabídne kuše či luk s ohnivými šípy, avšak pokud by používal oštěp, další vylepšení budou zas jiná. Každá zbraň má svou specifikaci, čímž se pro hráče naskýtá nespočet možných kombinací, jak hru hrát. Například kuše střílí velmi rychlé střely, avšak velmi pomalu se nabíjí, takže hráč je obecně zranitelnější, avšak nebezpečnější. Oproti tomu luk s ohnivými šípy střílí s větší frekvencí šípy, které sice zraňují méně než kuše, avšak po krátkou dobu zůstanou na místě hořet, čímž mohou ohrozit neopatrného protivníka.

Vylepšit zbraň lze maximálně třikrát (platí pro současný stav – je možné, že autor přidá nové úrovně zbraní), avšak ostatní vlastnosti lze posilovat teoreticky do nekonečna. Pro každé nové vylepšení je potřeba stále vyšší počet získaného skóre od předchozí úrovně, tím pádem bude vylepšování s další úrovní trvat déle.

Co se týče počítačových nepřátel, tak ti jsou ve srovnání s lidskými hráči naprogramování mnohem silnější, avšak (většinou) pomalejší. Lidské hráče převyšují v počtu životů a v síle své zbraně (která je unikátní pro konkrétního nepřítele – hráč ji získat nemůže). Pokud se hráč rozhodne utkat s počítačovým nepřítelem, bude to pro něj představovat opravdovou výzvu a vysoké riziko. Jako odměnu za poražení na druhou stranu obdrží relativně velký počet skóre. Počítačoví nepřátelé tu jsou implementováni jak z důvodu ozvláštnění celé hry, ale také jako další možný cíl pro hráčem, což je výhodné hlavně v momentě, kdy je na serveru nedostatek lidí a hráči nemají s kým soupeřit.

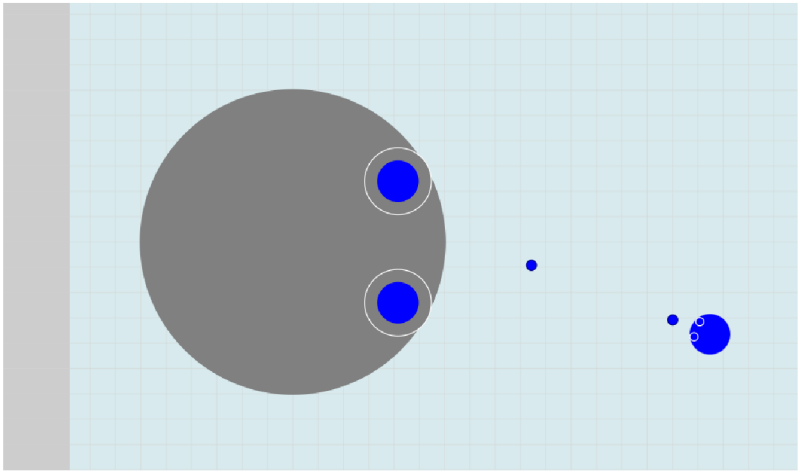
4.1.2 Vzhled hry

V této kapitole budou zobrazeny snímky ze hry, aby si čtenář mohl udělat představu o konkrétní podobě hry Fanior.



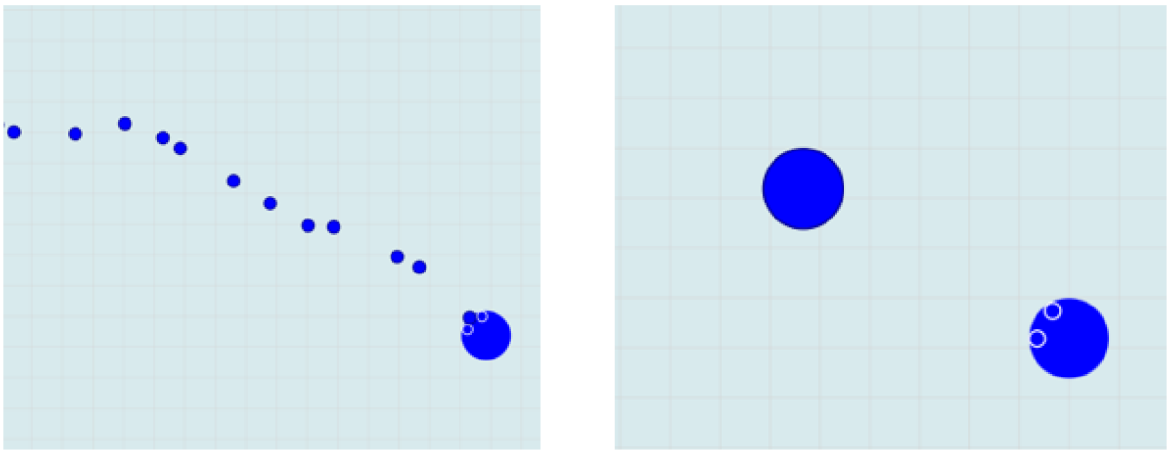
Obrázek 13: Střet dvou hráčů.

Postava ovládaná hráčem je modré barvy, nepřátele jsou červení.



Obrázek 14: Počítačový nepřítel

Počítačoví nepřátelé (tzv. guardians) se pomalu pohybují a ohrožují hráče v dosahu. Zdolání guardiana je opravdová výzva.



Obrázek 15: Porovnání zbraní

Zatímco nalevo hráč disponuje zbraní s rychlým přebíjením, napravo hráč soupeře ohrožuje velkými střelami, které téměř každého zdolají na jednu ránu – je jen na hráči, jakou strategii zvolí.

4.1.3 Předejití dominance jednoho hráče

Vylepšovat se lze teoreticky do nekonečna, nicméně nová úroveň vylepšení vyžaduje stále více skóre. Tento přístup je nebezpečný v tom, že v aréně může vzniknout hráč, který by byl příliš silný na to, aby ho kdokoliv porazil, což by mohlo vést k nehratelnosti, kdy by jakýkoliv nově připojený hráč byl velmi rychle zničen, protože nemá šanci obstát proti mnohokrát silnějšímu hráči.

Problém může být řešen více způsoby. Mezi nejběžnější mechanismy užívaných v online hrách podobného typu patří zpomalení pohybu hráče v závislosti na rostoucím skóre jako například ve hře diep.io (Aznatf 2018). Hráč, který je tedy silnější vzhledem k velkému počtu vylepšení, je zároveň pomalejší než nově připojení. Tím pádem má hráč na nízké úrovni stále šanci přežít, protože dokáže uniknout silnějším protihráčům.

Další přístup je přidání jednorázového štítu, který se na rozdíl od životů neobnovuje. Pokud je hráč zasažen, neuberou se mu životy, ale sníží se mu jeho štít. Teprve když je jeho štít zničen, začnou se mu po dalších zásahách ubírat životy. Tím, že se štít neobnoví, lze předpokládat, že hráči na vyšší úrovni (kteří jsou tím pádem pravděpodobně připojení déle) o štít již přišli. Naopak noví hráči disponují štítem, který jim zaručí větší šanci na přežití, než se vylepší na dostatečnou úroveň, aby mohli čelit ostatním hráčům. Tento přístup používá například hra takepoint.io.

Poslední zde zmíněné řešení problému dominance jednoho silného hráče přichází ze strany hráčů a ne vývojářů. Ve hrách tohoto typu si hráč samotný zvolí svůj cíl. Intuitivně chce každý získat co nejvyšší skóre. Kromě toho se však hráč samotný chce stát nejlepším v aréně. Toho dosáhne buďto tak, že svým skórem předčí současného leadera (tzn. hráče s nejvyšším skóre) nebo ho jednoduše zničí sám. Tímto způsobem se nejsilnější hráč stává terčem těch, kteří by se rádi stali leadrem. Navíc vývojáři toto mohou motivovat pomocí vedlejších úkolů tzv. achievementů. Achievementy spočívat ve zničení velkého počtu nepřátel, dosažení určité úrovně vylepšení a podobně. Pokud vývojáři vytvoří achievement s textem „Znič leadera arény“, bude pro dominantního hráče náročné udržet svou vůdčí pozici.

V demonstrativní hře je použitý přístup zpomalování se zvyšujícím se skóre. Kromě toho je zde použitý i systém neobnovitelného štítu pro nově příchozí hráče. Achievements ve hře zatím implementovány nejsou, nicméně je možné, že je autor v budoucnu do hry přidá.

4.2 Použitý framework

Celá hra Fanior byla vytvořena pomocí dříve zmíněného frameworku Blazor. Blazor je webový framework pro jazyk C# vycházející z frameworku .NET. Nejzákladnější volba pro tvorbu jakýchkoliv webových aplikací je po většinou jazyk JavaScript, na kterém je postavena řada frameworků (jako například React či Angular). Nicméně volba dynamicky typovaného jazyka jako je právě JavaScript nemusí být pro počítačovou hru výhodná. Ačkoliv dynamicky typované jazyky mají jisté výhody, tak zde by negativní aspekt převážil. Celá architektura hry je postavena na pevném objektovém návrhu s dědičností. Jasný a striktní design architektury programu udrží program čitelný, stabilní a lehce rozšiřitelný. V případě, že by na vývoji spolupracoval tým lidí v dynamicky typovaném jazyku, mohlo by snadno dojít k neporozumění částem kódu a následnému pádu programu (tomu samozřejmě lze předejít i v jazycích jako je JavaScript, nicméně silně typovaný je pro tyto potřeby vhodnější).

Blazor vzniknul v roce 2018, jedná se tedy o relativně novou technologii. Blazor ulehčuje vývoj webových aplikací, disponuje funkcemi pro snadnou online komunikace a nabízí programátorovi jasnou strukturu vhodnou pro vývoj online aplikací.

Stánky vytvořené Blazorem jsou tvořeny ze dvou základních komponent. První typ jsou soubory ve formátu .razor, které zastupují HTML stránku při standardním programování webových stránek. Zde je designován vzhled samotné stránky převážně pomocí HTML tagů. Kromě toho lze však HTML kód kombinovat s funkcemi, které nabízí jazyk C#. Pokud by například programátor chtěl stokrát vypsat text „Hello world“, stačil by napsat tento kód:

```
for (int i = 0; i < 100; i++)
{
        <p>Hello world</p>
}
```

V tomto souboru je také možné přímo pracovat s proměnnými z kódu v jiných souborech:

```
for (int i = 0; i < @legnth; i++)
{
        <p>@myText</p>
}
```

Druhý typ formátu souboru je .razor.cs, který obsahuje kód programu pro danou stránku v jazyce C#. Pomocí tohoto provázání HTML a samotného kódu je programování velmi usnadněno – pro změnu textu v odstavci na stránce není nutné zdlouhavě zjišťovat pomocí indexu daný element, následně mu nahrát nový text a stránku aktualizovat, jak by to bylo provedeno při běžné tvorbě webových stránek – v Blazoru lze užít proměnné přímo v kódu stránky.

Avšak ne všechny funkce jsou Blazorem přímo podporovány. Pro tyto případy se zde naskýtá možnost provázat aplikaci s JavaScriptem. Ku příkladu v samotném Blazoru neexistuje možnost pro zobrazení vyskakovacího okna v prohlížeči. Tuto funkci však JavaScript nabízí, proto není problém zavolat ze C# kódu funkci definovanou v JavaScriptu (vše je však oddělené pro celkovou čistotu kódu – volat funkce z JavaScriptu lze pomocí JSInterop, který zavolá požadovanou funkci, předá jí argumenty a následně vrátí výsledek funkce).

Blazor přichází s dvěma základními variantami – Blazor Server nebo Blazor WebAssembly. Kromě těchto dvou existuje ještě třetí možnost Blazor Hybrid, který bude také krátce popsán.

Blazor Server

Tento přístup funguje na principu neustálé výměny dat mezi klientem a serverem – do jisté míry Blazor Server přebírá architekturu autoritativního serveru.

Při počátečním dotazu klienta na server je dynamicky vytvořena HTML stránka, kterou server přepošle klientovi společně s potřebným CSS a JavaScriptem. Společně s tím je započato stálé spojení mezi serverem a klientem pomocí již popsané technologie SignalR. Veškerý kód hry běží čistě na serveru, který klienta neustále dynamicky aktualizuje (bez nutnosti stránku načítat). Přeposílány jsou pouze změny v HTML kódu, podle kterých klient vykresluje tuto stránku. Kód na straně klienta je tedy minimální – ve své podstatě zde běží pouze spojení se serverem. Jakákoliv logika programu není možná ke klientovi přidat.

Výhoda i nevýhoda je zde jasná – online komunikace je vyřešena a programátor se jí nemusí zabývat. SignalR je relativně stabilní, spolehlivá a rychlá technologie, tudíž se vývojář může spolehnout na to, že spojení mezi serverem a klienty je dostatečně kvalitní.

S tím se pojí také fakt, že přeposílané zprávy jsou velmi malé, jelikož program je spuštěn na serveru a klientovi tedy kód programu nemusí být vůbec poslán. To také řeší mnoho problémů spojených s bezpečností. Celkově Blazor Server těží z výhod autoritativního serveru, které jsou již popsány v předchozích kapitolách.

Hlavní nevýhoda však je, že programátor nemůže do komunikace jakkoliv zasáhnout. Klient je naprosto závislý na serveru. V případě výpadku je aplikace nepoužitelná. V momentě, kdy by programátor chtěl přidat jakékoliv funkce ke klientovi (spravování výpadku spojení apod.), není mu to v této variantě umožněno. Toto řeší následující přístup.

Blazor WebAssembly

Hlavní rozdíl oproti Blazor Serveru je ten, že kód je spuštěn jak u klienta, tak i na straně serveru. Zde je již umožněno programovat kód specifický pro server i pro klienta.

Vývojář musí sám zajistit spojení mezi serverem klientem. Je nechána volba na něm, zdali zvolí SignalR nebo se spokojí s HttpRequest či použije jakoukoliv jinou technologii.

Struktura programu je rozdělena na tři části: Client, Server a Shared.

Složky Client a Server jsou intuitivně jasné – v těchto složkách je tvořen kód zvlášť pro klienta a zvlášť pro server. Sekce Shared umožňuje přidat kód, který programátor chce mít přístupný od obou stran.

Pro tvorbu online hry je toto ideální. Veškerý kód hry může být spuštěn jak na serveru, tak i na zařízení hráče, což přímo nahrává algoritmům pro kompenzaci latence.

Výhoda oproti Blazor Serveru je tedy v možnosti přidat kód klientovi. Nevýhoda je v první řadě v nutnosti ručně ustálit spojení mezi serverem a klientem, v druhé řadě ve velikosti posílaných dat, které jsou oproti Blazor Serveru značně větší, což samozřejmě souvisí s dobou, za kterou se stránka prvně načte. Dalo by se tedy s nadsázkou tvrdit, že tento přístup je pro náročnější programátory, zatímco Blazor Server je vhodnější pro jednoduché aplikace (Pine 2022).

Blazor Hybrid

Blazor Hybrid užívá diametrálně odlišný přístup od předchozích zmíněných. Nejedná se již o aplikaci, která je spustitelná v prohlížeči. Blazor Hybrid umožňuje tvořit desktopovou či mobilní aplikaci, která pouze používá online spojení. Blazor Hybrid lze kombinovat s Windows Forms nebo s Windows Presentation Foundation, čímž vývojářům nabízí skvělý nástroj pro online spojení mezi serverem a klientem v desktopové aplikaci pro Windows (Latham et al. 2023).

4.3 Použité algoritmy a jejich efektivita

4.3.1 Objektový návrh demonstrativní hry

Nejzákladnější pilíř, na kterém stojí celá tato hra, je flexibilní objektový návrh.

Základní abstraktní třída je nazvána Item. Od této třídy dědí všechny ostatní objekty nacházející se ve hře. Tato třída obsahuje atributy, které jsou potřebné pro každý objekt ve hře. Jsou to souřadnice X a Y, identifikační číslo objektu, obalový objem a vzhled objektu, dále informace, zdali se jedná o neprostupný objekt či nikoliv (neprostupné objekty se nemohou navzájem překrývat. Zeď je například neprostupný objekt, střela není). Kromě těchto atributů obsahuje metody Collide sloužící pro vyhodnocení detekce kolize, Dispose, která se spustí při zničení objektu, a nakonec SetItem pro počáteční nastavení objektu. Z tohoto důvodu byla upřednostněna abstraktní třída před rozhraním.

Od této třídy dědí abstraktní třída Movable pro všechny objekty, u kterých je předpokládána možnost pohybu. Zde jsou přidány atributy a metody pro uskutečnění pohybu. Zvolený algoritmus pro pohyb spočívá prvně v nastavení, jakým způsobem se bude objekt pohybovat – zrychleně, rovnoměrně… Pokud by například objekt pohybující se rovnoměrně vstoupil

na led, může snadno změnit typ svého pohybu na zrychlený. Kromě toho objekt třídy Movable obsahuje dynamické pole instancí třídy Movement pro nové pohyby. Pokud například hráče odhodí výbuch, přidá se mezi jeho pohyby nový pohyb pro odhození (pravděpodobně by se jednalo o zrychlení určitým směrem), který bude započítán při vyhodnocování celkového pohybu. Kromě tohoto Movable poskytuje metody pro okamžité zastavení objektu SuddenStop (například při kolizi se zdí), pro postupné zastavení SmoothStop (v momentě, kdy hráč přestane držet klávesu pro pohyb) a pro přidání nového pohybu AddMovement (v případě, že na objekt začala působit nová síla).

Třídu Movable dědí abstraktní třída Character, která slouží jako šablona pro všechny postavy ve hře. Zde je nový atribut pro počet životů a pro typ zbraně. Druhá třída dědící Movable je abstraktní třída Shot, od které dále dědí třídy samotných výstřelů. Zde nalezneme atributy jako Damage (síla zranění) nebo CharacterId, podle kterého je určeno, která postava střelu vytvořila (aby nezranila sama sebe).

Od třídy Character dědí již ne abstraktní třída Player (hráč) či třída Enemy (počítačový nepřítel).

Zvlášť stojí třída Weapon, která tvoří šablonu pro všechny zbraně, které postavy mohou užívat. Tato třída nedědí od třídy Item, jelikož se nejedná přímo o objekt ve hře. Weapon obsahuje atributy jako je reloadTime určující frekvenci střelby či proměnnou autoFire, která určuje, zdali hráč může střílet neustále při držení klávesy nebo musí pro každé vystřelení stisknout klávesu znovu.

Pro realizaci pohybu byla vytvořena také již zmíněná abstraktní třída Movement. Právě z instancí potomků této třídy je složen výsledný pohyb hráče.

Pro celkovou kontrolu jedné arény byla vytvořena třída Gvars (zkratka ze slov game variables – herní proměnné). Pojmem aréna je nazýváno jedno herní prostředí či jedna místnost pro skupinu hráčů. V případě, že by se připojilo větší množství hráčů, byla by vytvořena nová aréna. Ve třídě Gvars je možné nalézt dynamická pole pro zaznamenání všech objektů ve hře, identifikační název arény, číselná proměnná Id, která slouží pro udělování identifikačních čísel jednotlivým objektům (s vytvořením nového objektu vzroste o jedna). Dále se tu nachází výška a šířka samotné arény (myšlenka je, že podle počtu hráčů se bude aréna dynamicky zmenšovat či zvětšovat – zatím neimplementováno).

Třída Gvars obsahuje všechny informace o dané aréně. Při připojení nového hráče mu tedy stačí přeposlat instanci této třídy pro konkrétní arénu, do které se má právě připojit.

Na serveru je dále implementována třída GameControl, ve které jsou informace týkající se kompletně celého serveru. Právě zde je proměnná messageId pro kontrolu výpadku zpráv (po každé poslané zprávě vzroste o jedna). Dále zde lze nalézt pole všech arén, metody pro komunikaci s klientem, metodu pro vytvoření nového hráče. V této metodě je také měřen čas v milisekundách od spuštění serveru (tento údaj není zatím využívaný, avšak může být užitečný pro budoucí vývoj).

Na straně klienta zas lze nalézt třídy pro kontrolu stisku kláves.

Ke každé klávese je přiřazena akce, která se se stiskem či puštěním klávesy vykoná. Název této akce je přeposlán serveru, který ji vyhodnotí.

4.3.2 Detekce kolize v demonstrativní hře

Grafika ve hře Fanior je pojata co nejjednodušeji. Cílem práce nebylo vytvořit krásnou a graficky propracovanou hru. Celá hra má cíleně působit dojmem jednoduchosti. Hra nevyžaduje pokročilý grafický procesor ani velkou paměť, kam by se uložily textury a obrázky. Vzhled většiny objektů tvoří pouhý olemovaný kruh. Tento přístup byl zvolen k co nejméně složité implementaci detekce kolize.

Každý objekt ve hře lze aproximovat na kruh. Pokud se tedy kontroluje srážka dvou objektů, stačí porovnat souřadnice středů a velikost poloměrů obou objektů.

Objektový návrh umožňuje přidat a následně doprogramovat objekty s jinými obalovými objemy, avšak implementován je zatím algoritmus pro objekty tvaru kruhu.

Na zlepšení efektivity celého programu byl použit quadtree. Vzhledem k relativně malé ploše a ne příliš velkému počtu objektů je celé herní pole pevně rozděleno na 16 čtverců, ve kterých jsou kontrolovány srážky objektů náležící do stejného čtverce. Pokud by došlo k dalšímu rozvoji hry a autor by usoudil, že je tento přístup již nedostatečný, nebyla by velká potíž přeprogramovat stávající pevně určený quadtree na dynamický, kdy by se plochy dělily sami podle počtu objektů v nich.

4.3.3 Online komunikace demonstrativní hry

Tato kapitola navazuje na teorii online komunikace popsanou v kapitole Online komunikace, Klient-server. Vytvořená hra právě využívá architekturu klient-server. Server zpracovává přijatá data a výsledky rozesílá klientům šedesátkrát za sekundu. Vzhledem k jednoduché grafice a celkové jednoduchosti hry byl zvolen snadnější přístup. Stejný kód hry běží na straně serveru i klienta. Klient však své akce neprovádí okamžitě, ale data nejprve pošle serveru, ten je sloučí s daty od všech uživatelů a v následujícím snímku přepošle seznam všech vykonaných akcí společně s indexem zprávy pro kontrolu, zdali nedošlo k výpadku. Nejedná se tedy čistě o autoritativní server s nečinnými klienty. Podle popsaného přístupu v kapitole Přenos informace, Algoritmy pro přenos dat, jsou přenášeny pouze předpřipravené příkazy. Pokud je tedy přijatý příkaz „moveUp“, posune se daný hráč směrem nahoru. Veškeré kolize, změny stavu a všechny algoritmy celkově jsou tedy prováděny jak u na straně serveru, tak i u klienta a spouštěny podle předdefinovaných názvů. Pokud klient zjistí, že zpráva nebyla doručena, požádá server o zaslání celkového stavu hry.

Tento přístup, ač ne dokonalý, je pro účely konkrétně této hry dostatečný.

Kromě toho se ve hře vyskytují algoritmy, které se odehrávají pouze na straně klienta nebo na straně serveru. U serveru je to již zmíněná komunikace se všemi klienty, kromě toho vytvoření hráče při zaznamenání nového přihlášení či vedení jednoduchých statistik (počet připojených hráčů a podobně). Na straně klienta je to zas úvodní přihlášení, interakce s ovládacími prvky hry (volba nové zbraně apod.), nastavení atd. Různé informace serveru ani posílány nejsou (to, jestli hráč změní nastavení kláves, server vědět nemusí).

Co se týče užitých technologií, pro online komunikaci je užita knihovna SignalR usnadňující implementaci funkcí pro přenos informace. Hlavní volbou SignalR je podpora použitého frameworku Blazor. Další výhoda této knihovny je, že udržuje na rozdíl od HttpRequest neustálé spojení s klientem, což přímo nahrává architektuře této online hry – předpokládá se delší doba spojení mezi konkrétním klientem a serverem.

Informace jsou přenášeny ve formátu JSON. Jak již bylo zmíněno v kapitolách rozebírající tyto formáty, vhodnější by bylo použít MessagePack. K obhájení volby JSON je nutné pohlédnout na programování z praktického hlediska. JSON byl zvolen jako známý a populární formát s kvalitní dokumentací a podporou pro většinu frameworků včetně

Blazoru. Na JSON autor narazil jako první a z hlediska implementace bylo relativně snadné JSON použít. Při následném porovnávání formátů autor zjistil, že formát MessagePack by byl z různých hledisek lehce lepší, nicméně čas strávený přeprogramováním z JSON na MessagePack by pro tento nepatrný posun za to nestál.

O samotný přenos dat se stará knihovna SignalR. SignalR přímo podporuje framework .NET, ve kterém je hra vyvíjena. Je navržena pro spojení, ve kterém se předpokládá častá komunikace a delší čas trvání – což je přesně vhodné pro online hry. Kromě toho SignalR podporuje jak JSON, tak i MessagePack. JSON je momentálně implementovaný ve hře. Pokud by ale došlo k potřebě přesunu na MessagePack, nenastane žádný problém (bradygaster 2023).

4.4 Optimalizační potenciál

Hra Fanior je plně funkční a hratelná, avšak i u sebelepší hry lze nalézt maličkosti, které by se mohly eliminovat. Fanior cílí na celkovou jednoduchost ať po stránce grafického designu, pravidel hry i použitých algoritmů. Zde vyvstává otázka, do jaké míry jsou algoritmy zvoleny vhodně. Při výběru algoritmů se autor rozhodoval podle následujících kritérií: rychlost, flexibilita a náročnost implementace. Algoritmus na vyhodnocování kolizí pomocí dynamického quadtree je sice rychlejší než přístup „každý s každým“, je také dostatečně flexibilní, protože lze teoreticky užít pro neomezený počet objektů na velkém herním poli. Avšak jeho implementace je náročnější s větším rizikem výskytu chyby v kódu. Ačkoli se může zdát čas strávený programováním jako nedostatečný argument pro výběr objektivně horšího algoritmu, praxe ukazuje jiné – pokud rozdíl mezi dvěma algoritmy je v realitě jednotky milisekund, není to pro vývojáře dostatečná motivace strávit programováním hodiny času, které může věnovat důležitějším oblastem hry.

Pro konkrétní představu zde bude uveden výčet současně použitých algoritmů společně s jejich potenciálně vhodnější verzí, která ve hře z již zmíněného důvodu použita není.

Problematika	Současný přístup	Vhodnější přístup
Obalový objem	Vše tvaru kruhu	Přidání možnosti výběru z více tvarů (např. obdélník)
Detekce kolize	Quadtree větvený vždy do 16 čtverců	Dynamicky větvený quadtree
Online komunikace	Klient neaktualizuje své akce okamžitě, ale čeká na odpověď od serveru.	Klient ihned vykoná akci a následně ji až porovná s odpovědí ze serveru.
Formát posílaných dat	JSON	MessagePack
Vyhodnocování akcí v čase	Jednotlivé akce jsou zařazeny do aktuálního snímku (jejichž frekvence je 60 za sekundu)	Akce jsou vyhodnocovány podle přesného času jejich vykonání.

Tabulka 1: Porovnání použitých a vhodnějších algoritmů

4.5 Postupný vývoj hry

V této kapitole bude nastíněn postup programování demonstrativní hry krok po kroku, vysvětleny obtíže v různých fázích vývoje, předloženy rady a inspirace pro čtenáře, kteří uvažují o programování online her i doporučený postup z osobního pohledu autora online hry. Obsah následující sekce je čerpán převážně z osobní zkušenosti, může se lišit od jiné literatury či od různých programátorů. Každý vývojář je jiný a každý může předat jinou zkušenost. Nicméně k inspiraci byla použita literatura Elements of Game Design (Zubek 2020) – s myšlenkami v této knize se autor ztotožňuje.

Typ hry

Nejprve je důležité stanovit si, o jaký typ hry půjde. Bude to strategická hra, příběhová, akční nebo jakákoli jiná? Budou hráči hrát spolu či proti sobě? Bude to propracovaná hra s krásnou grafikou, jejíž vývoj zabere týmu deseti lidí pět let, nebo se bude jednat o malou hru v hranou v prohlížeči, která nedá vlastně ani tolik práce vytvořit. Dále je potřeba určit, co bude cílem hry, proč to vlastně hráči budou hrát.

Dále vyvstává otázka, pro jakou skupinu je hra cílena? Jsou cíloví hráči desetileté děti s rozvinutým logickým myšlením nebo dospělí hráči s touhou vybít si agresi? Podle typu hráče se taktéž odvíjí i typ hry.

Obdobně jako malíři nejprve vytvoří náčrtek nebo spisovatelé osnovu příběhu, herní vývojář si taktéž musí vytvořit takovou osnovu, podle které bude tvořit svou hru.

Jakmile je představa o typu hry ujasněna, může se přejít k dalšímu kroku.

Příběh

Každá hra by měla mít svůj vlastní příběh – jakousi myšlenku, o co v dané hře vlastně jde, kdo je hráč v této hře a proč se vlastně snaží dosáhnout tohoto cíle. V šachách je hráč v roli generála snažícího se zdolat armádu nepřítele. Pokud by se figurky přejmenovaly ze střelce na postava 1 a z krále na postava 2, herní systém by to nijak neovlivnilo, ale popularita hry by jistě klesla.

I naprosto první hry měly svůj příběh – ve hře Tennis for two z poloviny 20. století se hráči ocitli v roli sportovce. Ve hře Space Invaders se zas snažili zachránit vesmír.

Před samotným programováním hry by měl mít tedy vývojář jasno, jaký je příběh jeho hry.

Pravidla

Nyní je třeba určit, jak konkrétně se hra bude hrát. Jakým způsobem budou hráči mezi sebou bojovat? Budou po sobě střílet nebo bojovat na blízko? Budou sbírat suroviny, ze kterých si něco mohou vytvořit anebo tam nebude žádný prvek strategie?

Po obecném návrhu hry je vhodné ujasnit si konkrétnější představu. Ku příkladu je rozhodnuto, že se jedná o jednoduchou hru, kde hráči po sobě střílí, postupně získávají skóre, za které si vylepšují své schopnosti. Následují nové otázky. Jaké schopnosti to budou? Mohou jich získat nekonečno, nebo se vývoj postavy na nějaké úrovni zastaví? Jakým způsobem bude hráč vyřazen ze hry? Chceme, aby se hráč léčil automaticky nebo pomocí jakýchsi lékárniček či jiných prvků? Budou ve hře počítačoví nepřátelé?

Jasnější návrh vždy pomůže postavit dobrou herní architekturu a nasměrovat programování správným směrem. A ačkoliv mnoho nápadů bude změněno a přidáno v průběhu vývoje, základy jsou již položeny.

Výběr frameworku

Vývoj hry velmi záleží na použité technologii. Špatná volba způsobí mnoho práce, času a energie navíc. Proto je zde doporučeno strávit dostatek času výběrem. Protože zjistit po půl

roce programování, že framework nakonec nevyhovuje požadavkům, se kterými se od začátku počítalo, může být zdrcující.

Blazor, který byl použitý pro vývoj hry Fanior, je z tohoto hlediska do jisté míry nevyhovující. Blazor není přímo herní framework – je to jen nástroj pro jednoduchou online komunikaci. Defaultně neobsahuje žádné prostředky pro vývoj her, veškeré algoritmy popsané výše si vývojář musí naprogramovat ručně (či použít jiné externí knihovny). Pro demonstrativní účely byl Blazor naopak vhodný přesně z toho důvodu, že bylo možné představit všechny postupy a algoritmy, jejich naprogramování a použití – v případě herních frameworků je většina těchto problémů pro vývojáře již vyřešena a ti jen volají příslušné funkce. To však nebylo cílem této práce.

Objektový návrh

Následuje vytvoření objektového návrhu. V některých aplikacích pro tvorbu her není ani možné tvořit objektové návrhy. Zde se však předpokládá použití takového programu, který umožňuje objektové programování.

S objektovým návrhem je možné se setkat i při vývoji jiných aplikací, než jsou hry. Avšak i zde je objektový návrh to, na čem stojí funkčnost a flexibilita celého programu. Kladen je důraz na co nejflexibilnější architekturu. Hlavně v případě hry obdobné Fanioru, kde se předpokládají různorodé zbraně, mnoho typů schopností a různých postav, je flexibilita fundamentální vlastností objektového návrhu. Pokud se vývojář rozhodne přidat nový typ zbraně, která náhodně mění rychlost přebíjení a vrhá rakety, které se nejprve zrychlují kupředu a při přiblížení k nepříteli se k němu obloukem stočí jako mířené střely, nebude to pro něj přílišný problém, jelikož jednoduše přidá novou zbraň s novými vlastnostmi a nemusí nijak zasáhnout do existujícího kódu.

Stejně tak jako u zbraní, tak i pro postavy ve hře nebo obecně pro všechny objekty může být vytvořena stabilní objektová architektura, takže nebude problém přidat jakýkoliv nový kód.

Základní logika

V této fázi je doporučeno vytvořit základy pro funkční a hratelnou hru alespoň pro jednoho hráče zatím bez online komunikace. V tomto kroku by vývojář měl být schopný vykreslit herní pole s postavou a několika objekty (zatím bez dokončené grafiky). Postava by měla

být schopna pohybovat se, interagovat s ostatními předměty (dotyk se zdí, zvednutí zbraně apod.).

Vývojář zde může pokračovat a dokončit veškeré algoritmy herní logiky, nicméně po začlenění komunikace s ostatními hráči může zjistit, že zapojení hry do online prostředí způsobuje nefunkčnost již naprogramovaného kódu. Proto je doporučeno vytvořit jen základ a přenést pozornost na online komunikaci.

Online komunikace

V této části vývoje programátor přidá možnost spojit se s ostatními hráči. Zde je doporučeno jít znovu od základů. Nejprve je třeba si uvědomit, jaká architektura bude použita – zda se serverem či bez něj. Dále naprogramovat odesílání a příjem dat od ostatních hráčů. Zatím není potřeba ladit menší nedostatky s latencí a implementovat složitější algoritmy, které by ji odstranily – základ stačí.

Vytrvalost

Poznatek z osobní zkušenosti: programování vyžaduje obrovskou dávku trpělivosti. Během programování se každý vývojář dostane do slepé uličky, stráví hodiny hledáním chyby v kódu, načež poté, co ji nalezne, program stále nefunguje. Nad algoritmem, který se zdá naprosto základní, najednou zůstane zaseknutý týden, a ačkoliv jeho aplikace včera ještě běžela, dnes se z neznámého důvodu nedá pustit. Online komunikace toto ještě umocňuje. Online hra naskýtá tolik situací, kde se může vynořit problém. Zde je jen na vývojáři, jak vytrvalý opravdu je. Pokud se však nenechá zaskočit nečekanými chybami a dokáže překlenout všechny potíže, se kterými se během programování jistě setká, tak nakonec vytvoří skvělou hru a dokáže tím své kvality.

Zde je vhodné zmínit jednu radu – stanovovat si dílčí cíle. Zaměřit se na jeden problém a ten vyřešit. Pokud se následující problematika může rozdělit, je vhodné zanalyzovat, co vše je potřeba a postupovat po jednotlivých krocích. Vždy je důležité práci odvést kvalitně, protože vracet se a opravovat nefunkční kód je mrhání časem, když ho lze napsat rovnou správně.

Potenciální problémy

Pro inspiraci zde budou uvedeno pár problémů a slepých uliček, se kterými se autor musel potýkat:

JSON nepodporuje delegáty, na kterých byla postavena celá architektura.

Kontrola stisku kláves se defaultně provádí tak často, že jakýkoliv jiný kód se při stisku klávesy téměř zastaví.

SignalR nedokáže přenášet různé datové typy, například vícedimenzionální pole – nijak však vývojáře nevaruje (jen tím, že aplikace nefunguje).

Nulový bod souřadnic je pro elementy vlevo nahoře – zvětšování vertikální složky tedy znamená posun dolu (ačkoliv běžná matematika pracuje s kladným směrem vzhůru).

Vícevláknové programování je zde základ – celá hra musí běžet, zatímco klient komunikuje se serverem a uživatel zadává vstupy. Více vláken přistupuje k týmž proměnným, což potenciálně může být problematické. Celý program je tedy nutné udržet thread-safe (tzn. zabezpečit, aby pouze jedno vlákno mohlo měnit data v konkrétní čas).

Vrátit se k rozdělaném kódu po dvou týdnech bylo náročné – vždy trvalo hodiny zorientovat se a navázat na předchozí práci. Komentáře jsou v této situaci velkým pomocníkem – až nyní autorovi došlo, že opravdu mají smysl. Kromě toho je vhodné založit si zápisník, kam si může vývojář zaznamenávat různé návrhy a myšlenky.

Tím, že je Blazor relativně nová technologie, neexistuje příliš mnoho tutoriálů nebo článků zabývajících se specifickými problémy. Vývojář se nesmí bát obrátit se na své kolegy na online fórech.

Střídavý vývoj

Po dokončení předchozích fází by měl být hotový základ hry. Ke hře se může připojit více hráčů, mohou se pohybovat a interagovat mezi sebou. K dokončení hry je však potřeba stále mnoho, a proto je doporučeno střídat pozornost mezi jednotlivými částmi. Nejprve například vyladit komunikaci mezi hráči a odstranit přílišnou latenci. Následně dokončit ovládání hráče, použít náročnější algoritmy na detekci kolize. Dále přidat uživatelské rozhraní – herní menu, okna pro výběr zbraní a jiné.

V této fázi je prospěšné nechat si hru kontrolovat od jiných lidí, kteří si hru zahrají a poskytnou zpětnou vazbu.

Cíl je hráč

Neustále je vhodné mít na paměti, že hra je programována pro hráče, který si má hru užít a při nejlepším se k ní později zas vrátit. Počítačová hra neslouží pro matematiky, kteří se snaží vyřešit algoritmizační problémy ani pro umělce, kteří cílí na co nejhezčí grafický design. Pokud má špatná hra skvělé algoritmy a nádhernou grafiku, stále to je špatná hra. Zábavné hry byly tvořeny i v době nevýkonných procesorů a nedokonalých algoritmů s naprosto základní grafikou – a i přesto je lidé s oblibou hráli.

Grafika a jiné efekty

Ke konci vývoje je možné přidat rozličné grafické prvky, animace, zkrášlit herní menu, protože přeci jen to je to první, s čím se hráč setká. Také se nabízí přidat zvukové efekty či hudbu do pozadí.

Závěrečné ladění

Ačkoliv je hra de facto hotová, je potřeba vyladit veškeré maličkosti, kterým nebyla v průběhu vývoje věnována pozornost. Například to může být vyvážení síly zbraní. Častým důvodem nespokojenosti hráčů jsou nevyvážené zbraně či schopnosti. Více zbraní do hry vnáší variabilitu, jak hru hrát, a proto si hráč může pokaždé zvolit jinou zbraň a skončit hru zas s jiným zážitkem než z minulé. Nicméně každá nová zbraň i každá nová schopnost musí být vyvážená, aby mohla čelit ostatním. Pokud si hráč zvolí zbraň, která je objektivně slabší než jiná, může být zklamán z prohry, která vlastně nebyla jeho vinou, ale vinou vývojáře, který nedokázal zbraně vybalancovat.

Při dokončování hry je přirozené mít tendence přidávat stále více a více nových prvků. To samo o sobě není škodlivé, avšak stále je potřebné udržovat hru funkční v každém novém kroku a veškeré další změny činit zvolna, a to zvláště v případě, kdy je hra již spuštěna.

Neopustit hru

Tip na úplný konec: I v momentě, kdy je hra hotova, má svou vlastní komunitu pravidelných hráčů, funguje bez chyb a nic zásadního již vývojář nehodlá měnit, je třeba hru postupem času aktualizovat. Ačkoliv se zdá, že hráči jsou spokojeni se stavem, v jakém se hra právě nachází, pravidelné (ale i nepravidelné a nečekané) aktualizace pomohou udržet loajální hráče motivované hrát hru i nadále.

4.6 Zdrojový kód hry

Na kompletní kód hry je možné nahlédnout na webové stránce GitHub pod profilem autora. Konkrétně zde: https://github.com/LinhartV/Fanior

Momentálně hra není spuštěna na žádném serveru. Je však možné, že v budoucnu autor nahraje Fanior na server a bude si ji moci kdokoliv zahrát.

5 Závěr

Cílem práce bylo vytvořit online kompetitivní hru s prvky nepřátel ovládaných počítačem.

Hra byla vytvořena ve webovém frameworku Blazor v jazyce C#. Jedná se o jednoduchou akční hru pro více hráčů, kteří spolu a počítačovými soupeři zápasí v reálném čase. Ve hře jsou použity algoritmy a postupy, o kterých pojednává právě tato práce. Je zde zhodnocen optimalizační potenciál hry, porovnány ideální algoritmy a algoritmy prakticky užité ve hře.

V první části této práce je rozebrán obecný kontext videoher, historie počítačových her, druhy videoher, dnešní trendy a dopady na hráče.

Druhá část se zabývá problematikou vývoje online hry. Jsou zde uvedeny postupy pro detekci kolize, online komunikaci, taktéž jsou nastíněny úskalí spojené s podváděním a jejich možná řešení. Dále jsou zde popsány technologie pro tvorbu online her – frameworky a herní enginy, které usnadňují programátorům vývoj počítačových her.

Třetí část uvádí demonstrativní hru. Uveden je objektový návrh hry a použité algoritmy, které byly ve druhé části teoreticky popsány. Čtenář se zde setká s tipy a radami týkajících se tvorby online hry, jak úspěšně vytvořit hru a dovést ji ke zdárnému konci. V posledních kapitolách jsou popsány osobní zkušenosti při tvorbě této hry, potíže a úskalí, se kterými se autor musel potýkat. Nakonec je uveden odkaz na kód samotné hry, kde čtenář může pohlédnout na konkrétní implementaci popsaných algoritmů.

Seznam zdrojů

ADAIR, Cam, 2021. The Negative Effects of Video Games - 12 Symptoms. *Game Quitters* [online] [vid. 2023-05-06]. Dostupné z: https://gamequitters.com/negative-effects-of-video-games/

AJ GLASSER, 2010. *Real money trading — the black market of online gaming | IT Business* [online] [vid. 2023-04-17]. Dostupné z: https://www.itbusiness.ca/news/real-money-trading-the-black-market-of-online-gaming/14774

ASSUNÇÃO, Carina, 2016. „No Girls on the Internet": The Experience of Female Gamers in the Masculine Space of Violent Gaming. *Press Start*. **3**(1), 46–65. ISSN 2055-8198.

AZNATF, 2018. Motion Mechanics #1 (Movement Speed). In: *r/Diepio* [online]. Reddit Post. [vid. 2023-03-23]. Dostupné z: www.reddit.com/r/Diepio/comments/9i9q1a/motion_mechanics_1_movement_speed/

BRADYGASTER, 2023. *Overview of ASP.NET Core SignalR* [online] [vid. 2023-05-08]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction

BROOKHAVEN NATIONAL LABORATORY, 2013. *Tennis_For_Two_on_a_DuMont_Lab_Oscilloscope_Type_304-A.jpg (705×626)* [online] [vid. 2023-05-07]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/5/50/Tennis_For_Two_on_a_DuMont_Lab_Oscilloscope_Type_304-A.jpg

BUREAU, Michelle, Edouard HIRSCH a Federico VIGEVANO, 2004. Epilepsy and Videogames. *Epilepsia* [online]. **45**(s1), 24–26. ISSN 1528-1167. Dostupné z: doi:10.1111/j.0013-9580.2004.451003.x

BURKE, Steve, 2015. *The Witcher 3's $81 Million Budget & Sales Figures Breakdown* [online] [vid. 2023-05-07]. Dostupné z: https://www.gamersnexus.net/industry/2083-the-witcher-3-budget-breakdown

CATHERINE LEWIS, 2022. New Survey Finds How Much The Average Gamer Spends On Gaming In Their Life. *GAMINGbible* [online] [vid. 2023-05-07]. Dostupné z: https://www.gamingbible.com/news/survey-reveals-how-much-gamers-spend-on-gaming-20220426

ČEJKA, Jan, 2016. *text_rigo_cejka.pdf* [online] [vid. 2023-05-07]. Dostupné z: https://is.muni.cz/th/wzqdb/text_rigo_cejka.pdf

DAVID S. COHEN, 2019a. Cathode-Ray Tube Amusement Device: The World's First Video Game? *Lifewire* [online] [vid. 2023-05-07]. Dostupné z: https://www.lifewire.com/cathode-ray-tube-amusement-device-729579

DAVID S. COHEN, 2019b. OXO aka Noughts and Crosses - The First Video Game. *Lifewire* [online] [vid. 2023-05-07]. Dostupné z: https://www.lifewire.com/oxo-aka-noughts-and-crosses-729624

ENTERTAINMENT SOFTWARE ASSOCIATION, 2022. U.S. gamers by gender 2022. *Statista* [online] [vid. 2023-05-06]. Dostupné z: https://www.statista.com/statistics/232383/gender-split-of-us-computer-and-video-gamers/

ERICSON, Christer, 2004. *Real-Time Collision Detection* [online]. 0 vyd. B.m.: CRC Press [vid. 2023-04-15]. ISBN 978-0-08-047414-4. Dostupné z: doi:10.1201/b14581

ERIN HAMILTON, 2009. *GameSpot - /gamespot/features/all/gamespotting/102702/8.html* [online] [vid. 2023-05-07]. Dostupné z: https://web.archive.org/web/20090211101758/http://www.gamespot.com/gamespot/features/all/gamespotting/102702/8.html

FISCHER, Peter, Jörg KUBITZKI, Stephanie GUTER a Dieter FREY, 2007. Virtual driving and risk taking: Do racing games increase risk-taking cognitions, affect, and behaviors? *Journal of Experimental Psychology: Applied* [online]. **13**, 22–31. ISSN 1939-2192. Dostupné z: doi:10.1037/1076-898X.13.1.22

GAMBETTA, Gabriel, 2022. *Client-Server Game Architecture - Gabriel Gambetta* [online] [vid. 2023-03-22]. Dostupné z: https://www.gabrielgambetta.com/client-server-game-architecture.html

GREENE, Tristan, 2019. Cyberpunk 2077's character creation system won't have gender options. *TNW | Gaming* [online] [vid. 2023-05-07]. Dostupné z: https://thenextweb.com/news/cyberpunk-2077s-character-creation-system-wont-have-gender-options

GREENFIELD, Patricia M., Craig BRANNON a David LOHR, 1994. Two-dimensional representation of movement through three- dimensional space: The role of video game expertise. *Journal of Applied Developmental Psychology* [online]. **15**(1), 87–103. ISSN 0193-3973. Dostupné z: doi:10.1016/0193-3973(94)90007-8

GROVES, Christopher a Craig ANDERSON, 2017. Negative Effects of Video Game Play. In: [online]. Dostupné z: doi:10.1007/978-981-4560-50-4_13

HALBROOK, Yemaya J., Aisling T. O'DONNELL a Rachel M. MSETFI, 2019. When and How Video Games Can Be Good: A Review of the Positive Effects of Video Games on Well-Being. *Perspectives on Psychological Science* [online]. **14**(6), 1096–1104. ISSN 1745-6916. Dostupné z: doi:10.1177/1745691619863807

HARVEY M. DEITEL a BARBARA DEITEL, 1986. *Machine Language Program - an overview | ScienceDirect Topics* [online] [vid. 2023-05-16]. Dostupné z: https://www.sciencedirect.com/topics/engineering/machine-language-program

CHILDCARE.CO.UK, 2023. Half of parents allow young kids to play 18+ games unsupervised. *Childcare.co.uk* [online] [vid. 2023-05-06]. Dostupné z: https://www.childcare.co.uk/blog/video-games

IVORY, James D., 2015. A Brief History of Video Games. In: *The Video Game Debate*. B.m.: Routledge. ISBN 978-1-315-73649-5.

JEAN-LOUP GAILLY, 2022. *GNU Gzip* [online] [vid. 2023-04-04]. Dostupné z: https://www.gnu.org/software/gzip/manual/gzip.html

JESSICA CLEMENT, 2022. Topic: Video game industry. *Statista* [online] [vid. 2023-05-07]. Dostupné z: https://www.statista.com/topics/868/video-games/

JESSICA CLEMENT, 2023. Top video game genres worldwide by age 2022. *Statista* [online] [vid. 2023-05-07]. Dostupné z: https://www.statista.com/statistics/1263585/top-video-game-genres-worldwide-by-age/

JOHN J. ANDERSON, 1983. *Features: Who Really Invented The Video Game?* [online] [vid. 2023-05-07]. Dostupné z: https://www.atarimagazines.com/cva/v1n1/inventedgames.php

JORDAN, Amy Beth a Daniel ROMER, 2014. *Media and the Well-being of Children and Adolescents*. B.m.: Oxford University Press. ISBN 978-0-19-998746-7.

JOUNI SMED a HARRI HAKONEN, 2006. *Preventing Look-Ahead Cheating with Active Objects*. 2006.

LABS, Malwarebytes, 2021. Gamers beware: The risks of Real Money Trading (RMT) explained. *Malwarebytes* [online] [vid. 2023-04-17]. Dostupné z: https://www.malwarebytes.com/blog/news/2021/09/gamers-beware-the-risks-of-real-money-trading-rmt-explained

LATHAM, Luke, Rick ANDERSON a Chris SAINTY, 2023. *ASP.NET Core Blazor Hybrid* [online] [vid. 2023-05-16]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/blazor/hybrid/

LEVELCAPGAMING, 2017. *maxresdefault.jpg (1280×720)* [online]. 6. srpen 2017. [vid. 2023-05-06]. Dostupné z: https://i.ytimg.com/vi/zFxzg6wviGg/maxresdefault.jpg

MDN CONTRIBUTORS, 2023. *Working with JSON - Learn web development | MDN* [online] [vid. 2023-04-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

MELISSA T. BUELOW, BRADLEY M. OKDIE, a ASHLEY B. COOPER, 2015. *The influence of video games on executive functions in college students | Elsevier Enhanced Reader* [online] [vid. 2023-04-19]. Dostupné z: doi:10.1016/j.chb.2014.12.029

NAMCO LIMITED, 1980. *Pac-man.png (224×288)* [online] [vid. 2023-04-15]. Dostupné z: https://upload.wikimedia.org/wikipedia/en/5/59/Pac-man.png

NÍCKOLAS DA SILVA, 2020. *Comparing JSON and MessagePack* [online] [vid. 2023-04-04]. Dostupné z: https://thephp.website/en/issue/messagepack-vs-json-benchmark

NOEMÍ, Peña-Miguel a Sedano Hoyuelos MÁXIMO, 2014. Educational Games for Learning. *Universal Journal of Educational Research* [online]. **2**(3), 230–238. ISSN 2332-3205, 2332-3213. Dostupné z: doi:10.13189/ujer.2014.020305

O'SHEA, Elizabeth, 2015. 10 Reasons Children Shouldn't Play Adult Rated Video Games. *Parent 4 Success* [online]. [vid. 2023-05-06]. Dostupné z: https://www.parent4success.com/2015/12/08/10-reasons-children-shouldnt-play-adult-rated-video-games/

PATHFINDER DIGITAL LLC, 2020. *Bit-Error Rate (BER) - PathFinder Digital Wiki* [online] [vid. 2023-05-08]. Dostupné z: https://wiki.pathfinderdigital.com/wiki/bit-error-rate-ber/

PETR FOJTÍK, 2020. Virtuální fotbalové derby Zlína a Slovácka pomůže zdravotníkům. *iDNES.cz* [online] [vid. 2023-05-06]. Dostupné z: https://www.idnes.cz/fotbal/prvni-liga/zlin-slovacko-virtualni-derby-pomoc-zdravotnikum.A200424_130748_fotbal_par

PINE, David, 2022. *Learning Blazor* [online] [vid. 2023-05-15]. Dostupné z: https://davidpine.net/blog/learning-blazor/

PRENSKY, Marc, 2003. Digital game-based learning. *Computers in Entertainment* [online]. **1**(1), 21–21. ISSN 1544-3574. Dostupné z: doi:10.1145/950566.950596

RHEO, 2015. *apple-985513_960_720.jpg (960×519)* [online]. 2015. [vid. 2023-04-15]. Dostupné z: https://cdn.pixabay.com/photo/2015/10/13/03/05/apple-985513_960_720.jpg

SEVERIN, Karol, 2022. The average gamer plays more than one hour per day, as time spent takes centre stage. *MIDiA Research* [online] [vid. 2023-04-19]. Dostupné z: https://www.midiaresearch.com/blog/the-average-gamer-plays-more-than-one-hour-per-day-as-time-spent-takes-centre-stage

STUART, Keith, 2021. *Medal of dishonour: why do so many people cheat in online video games? | Games | The Guardian* [online] [vid. 2023-03-28]. Dostupné z: https://www.theguardian.com/games/2021/feb/24/medal-of-dishonour-cheating-video-games

YAGER DEVELOPMENT GMBH, 2023. *Prospectors Code | The Cycle Frontier* [online] [vid. 2023-04-19]. Dostupné z: https://thecycle.game/prospectors-code

YAN, Jeff a Brian RANDELL, 2005. A systematic classification of cheating in online games. In: *NETGAMES05: 4th Workshop on Network and System Support for Games*: *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games* [online]. Hawthorne NY: ACM, s. 1–9 [vid. 2023-04-17]. ISBN 978-1-59593-156-6. Dostupné z: doi:10.1145/1103599.1103606

ZHOU, Daniel, 2018. Javascript: Javascript Dynamic Typing, Coercion and Operators. *Medium* [online] [vid. 2023-05-16]. Dostupné

z: https://medium.com/@easyexpresssoft/dynamic-typing-coercion-and-operators-a8986be8c198

ZUBEK, Robert, 2020. *Elements of Game Design*. B.m.: MIT Press. ISBN 978-0-262-04391-5.