Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Implementace síťové verze hry Monopoly Gamer

2023

Yevhenii Rubanskyi

Vedoucí práce:
doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní program: Informatika, prezenční forma

Bibliografické údaje

Autor:	Yevhenii Rubanskyi
Název práce:	Implementace síťové verze hry Monopoly Gamer
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informatika, prezenční forma
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	28
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Yevhenii Rubanskyi
Title:	Implement the network version of Monopoly Gamer
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Computer Science, full-time form
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	28
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Byla implementována síťová verze hry Monopoly Gamer, která umožňuje komunikaci uživatelů skrz síť v rámci jedné aplikace pomocí připojení k serveru. Vytvořeny jsou dva programy – hra a server. Hra byla vytvořena pomocí Unity, server je konzolová aplikace, která využívá technologie .Net6. Komunikace mezi klientem, samotnou hrou a serverem probíhá pomocí protokolu TCP.

Synopsis

A network version of the Monopoly Gamer game has been implemented, which allows communication between users through a network within a single application by connecting to a server. Two programs have been created - the game and the server. The game was created using Unity, and the server is a console application that uses .Net6 technology. Communication between the client, the game itself, and the server takes place using the TCP protocol.

Klíčová slova: TCP; UDP; Unity; implementace; C#; Monopoly

Keywords: TCP; UDP; Unity; implementation; C#; Monopoly

Děkuji vedoucímu mé práce doc. RNDr. Miroslavu Kolaříkovi, Ph.D. za pochopení, pomoc a rychlou spolupráci. Také bych chtěl poděkovat svým blízkým a rodině za podporu. Děkuji.

Odevzdáním tohoto textu jeho autor místopřísežně prohlašuje, že celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

1 Úvod

Implementace síťové verze hry, je napsaná pomoci enginu Unity a pro komunikaci využívá TCP protokol. Hra umožňuje připojení více hráčů současně a zajišťuje komunikaci mezi klienty a serverem. Během implementace jsem se zabýval návrhem a implementací síťového protokolu, vytvořením serverové aplikace v jazyce C# a implementací klienta hry v Unity.

Serverová část je konzolový program, který slouží pouze ke spojení klientů a komunikaci mezi nimi. Server je napsán s využitím .Net6, vybral jsem ho, aby bylo možné hotový program spustit na libovolném operačním systému.

Rozhodl jsem se zvolit jako téma své bakalářské práce implementaci síťové verze hry. Hlavním důvodem pro tento výběr byl můj zájem o vývoj her a programování sítí. Chtěl jsem lépe porozumět tomu, jak se vytváří multiplayerová hra a jakým způsobem jsou v této oblasti řešeny problémy sítí. Doufám, že tato práce mi umožní rozšířit své znalosti a dovednosti v této oblasti a zlepší mé šance na trhu práce.

2 Pravidla hry

Hlavním cílem deskové hry Monopoly Gamer je získat co nejvíce bodů, které se získávají nákupem a prodejem vlastností na herní desce a za poražené bosse.

Herní deska je rozdělena na políčky, které představují různé lokace ve světě Super Mario Bros. Hráči postupně házejí kostkou a pohybují se po polích, na kterých mohou nakupovat nemovitost a v závislosti na tom, o jaké políčko se jedná, vykonat jeho akci.

Hráči si mohou vybrat jednu z čtyř herních figurek z nejpopulárnějších postav ze světa Super Mario Bros. Každá figura má své unikátní schopnosti, které mohou hráčům pomoci v jejich úsilí získat více bodů.

Hra také obsahuje speciální kostku s herními schopnostmi, které mohou hráči použít k ovlivnění hry a získání výhody. Cílem hry je získat co nejvíce bodů a stát se nejbohatším hráčem na konci hry. Maximální počet hráčů jsou 4 osoby.

2.1 Základní pravidla hry

Herní pole se liší tím, že místo nemovitostí a ulic, které jsou základem klasické hry Monopoly, jsou místa a lokace, které se objevují ve světě Super Mario, které hráči mohou kupovat a prodávat.

Na herním poli jsou nová políčka:

- Thwomp,
- Coin Block,
- Super Star Ability,
- Warp Pipe.

Každé políčko má svou funkci, například pokud se dostanete na Warp Pipe, vaše figurka se automaticky přesune na místo dalšího políčka Warp Pipe. Coin Block – pokud se dostanete sem, musíte hodit kostkou s čísly a získáte počet mincí odpovídající číslu na kostce. Thwomp – musíte dát 2 mince na toto políčko. Super Star Ability – aktivuje super schopnost vaší figurky.

Kromě toho je další odlišnost od klasické hry v tom, že místo dvou kostek s čísly, od jedné do šesti, používáme jednu kostku s čísly, od jedné do šesti, a kostku, na níž jsou obrázky posilovačů (Power-Up). Zobrazení na druhé kostce jsou následující: Red Shell, Green Shell, Blooper, Coins, POW. Jak můžeme vidět, bylo zde vypsáno celkem pět posilovačů. Důvodem toho je, že jeden prvek se opakuje, a to je mince (Coins), které jsou umístěny naproti sobě na kostce. Proto seznam, který jsem uvedl, se skládá pouze ze pěti prvků.

Pokud vám padne:

- Red Shell, tehdy vyberete jednoho z hráčů, kromě sebe, vybraný hráč sází 3 mince na pole, kde se nachází.

- Green Shell – je podobný Red Shellu, ale liší se tím, že vybíráte pouze hráče před sebou, pokud se na jednom poli nachází více hráčů, vyberete jednoho.
- Blooper – vezmete si od libovolného hráče 2 mince.
- Coins – získáte 3 mince.
- POW – všichni kromě vás sází na místo, kde se nacházejí, 1 minci.

Kromě super schopností mají figurky také unikátní posilovače, například Mario má posilovač pro Coins, místo 3 mincí získá 4 mince.

Ve hře Monopoly Gamer se setkáváme s vězením, parkováním zdarma, odesíláním do vězení a startovním polem, s čím se můžeme potkat i v klasické hře Monopoly. Také platí pravidlo, že pokud přejdete startovní pole, získáte 2 mince. Ale Monopoly Gamer se liší i tím, že když přejdete startovní pole, začínáte boj s bossem (Boss Fight), který začíná tím, že otočíte kartu bosse, která se nachází uprostřed herního pole.

2.2 Hlavní rozdíl – boj s bossem

Stejně jako figurky hráčů mají i bossové své schopnosti, každý boss má minimální číslo, které je třeba dosáhnout, abyste ho porazili, počet mincí, které hráč musí zaplatit, aby začal boj, super schopnost, která se aktivuje, když hráč bosse porazí, a počet mincí na kartě.

Boj probíhá tak, že hráč zaplatí určitý počet mincí uvedených na kartě bosse, aby s ním začal boj, poté hodí kostkou s čísly, pokud číslo na kostce je větší nebo rovno číslu uvedenému na kartě, aby zvítězil, tak hráč vyhrává a aktivuje se schopnost bosse. Jestliže číslo je menší, tehdy hráč prohrává a boj s bossem přechází na dalšího hráče.

Pokud hráč nechce bojovat, může boj přeskočit a ukončit svůj tah, v tom okamžiku se tah přesouvá na dalšího hráče, který rozhodne, zda bude bojovat nebo ne, pokud i tento hráč přeskočí boj, pak přichází na řadu další hráč. Pokud všichni hráči přeskočili bitvu, karta bosse opouští hru. V případě vítězství nad bossem hráč získá jeho kartu. Celkově je v hře 8 bossů, po boji s posledním bossem hra končí.

2.3 Zvláštnost políčka nemovitostí

Stejně jako v klasické hře Monopoly, když hráč přistane na volné políčko nemovitosti, může ji koupit, nebo pokud mu chybí mince na její nákup, vystavit ji na aukci.

Při aukci je počáteční sázka stanovena na jednu minci. Po počáteční sázce mohou hráči zvýšit svou sázku pouze o jednu minci. Hráči také mohou odmítnout účast na aukci a zvýšení sázky. Aukci vyhrává hráč, který zůstane poslední a platí bance tolik mincí, kolik odpovídá poslední sázce aukce.

V případě, že hráč přistane na nemovitostní pozici, kterou již někdo vlastní, musí zaplatit určitý počet mincí majiteli pozice. Konečná částka závisí na tom, zda hráč vlastní celou sadu této nemovitosti, sadou se rozumí nemovitostní políčka stejné barvy. Pokud nemá dostatek mincí, musí prodat jednu nebo více karet nemovitostí, které vlastní. Pokud nemá ani mince, ani vlastnictví, nic nemůže udělat.

Příjem mincí může být získán následujícími způsoby: když ostatní hráči přistoupí na pole nemovitosti, kterou vlastníte, platí vám nájemné, při přechodu pole Start, když se na kostce objeví posilovač Coins, když přistoupíte na pole Coin Block nebo prostě když si je vezmete z polí, kterými projdete při svém tahu.

2.4 Konec hry

Vítězí hráč s nejvíce body (Points). Počítání bodů hráčů funguje tak, že se spočítá počet mincí, nejvyšší číslo dělitelné beze zbytku 5, následně se tato hodnota vynásobí 2, sečte se s počtem bodů z poražených bossů a sečte se s celkovým počtem bodů ze všech nemovitostí, kterými hráč disponuje.

2.5 Pravidla hry, která nebyla implementována.

Několik slov o tom, co ve verzi mé hry není. Hráč nemůže prodat svůj majetek jinému hráči a nelze zvolit typ hry, který závisí na počtu hráčů.

3 Technologie

Moje práce je napsána v programovacím jazyce C#. Dále jsem použil technologie Unity, protokoly TCP a UDP, výměna informací probíhá odesíláním textu ve formátu string, kde část textu je v JSON formátu. Rozhodl jsem se nepřidávat žádnou databázi, protože tato hra je krátká a neobsahuje mnoho informací, které by bylo potřeba ukládat. Serverová část hry se skládá z jednoho programu – konzolové aplikace. Server vystupuje v roli spojení mezi klienty, výměnou a zpracování informací mezi nimi. Pro klientskou část, tedy samotnou hru, jsem navrhl svůj vlastní design. Navrhoval jsem ho na webu Figma.com[1], abych měl představu o tom, co budu vytvářet. Po vytvoření designu jsem začal v Unity vytvářet uživatelské rozhraní.

3.1 Unity

Unity jsem si vybral proto, že za prvé mohu psát logiku pro hru v jazyce C#, za druhé má poměrně snadný konstruktor a bohatou funkcionalitu, za třetí na oficiálních stránkách vývojářů existuje mnoho dokumentace o použití jejich programu.

3.2 TCP

Základem komunikace klient-server je protokol TCP, vybral jsem ho, protože zajišťuje připojení k serveru a garantuje odesílání dat.

3.3 UDP

Protokol UDP používám pouze jednou, a to tehdy, když uživatel neví adresu serveru. Potom pomocí tohoto protokolu odesílám pakety na IP adresy v rozsahu 192.168.1.1-254. Tento rozsah jsem zvolil jen proto, že většina domácích sítí má takový standardně nastavený rozsah. Ale rozhodl jsem se ho nepoužít jako základ pro komunikaci, kvůli jeho nedostatkům: nezaručuje odeslání nebo přijetí odeslaných informací (datagramů).

3.4 Implementace TCP v jazyce C#

Rozhodl jsem se použít TcpListener místo Socket kvůli jeho jednoduššímu použití a vyšší úrovni abstrakce, což usnadňuje tvorbu serverové aplikace. TcpListener poskytuje vyšší úroveň rozhraní pro síťové programování a umožňuje jednodušší a srozumitelnější implementaci síťového spojení mezi klientem a serverem. TcpListener automaticky přijímá příchozí připojení a umožňuje snadnější zpracování různých situací, jako je připojení nového klienta, odpojení klienta. Což dělá kód více přehledným.

[1] figma.com

Tím, že jsem zvolil TcpListener místo Socket, zjednodušil jsem proces vývoje a umožnil jsem si rychlejší a efektivnější vývoj serverové a klientské části hry.

3.5 JSON

Pro komunikace mezi serverem a klientem používám formát JSON pro přenos dat. JSON je textový formát pro výměnu dat, který umožňuje pohodlný a jednoduchý přenos strukturovaných dat mezi různými systémy. Každý JSON objekt se skládá z dvojice „klíč-hodnota“, kde klíč je řetězec a hodnota může být libovolný typ dat, včetně pole a jiných JSON objektů. Pro serializaci a deserializaci objektů na formát JSON používám standardní knihovnu pro serializaci System.Text.Json v C# .

4 Postup vývoje

Jak jsem již psal výše, nejdříve jsem navrhl design programu, abych měl představu, jak a s čím budu pracovat. K tomu jsem použil webovou stránku Figma.com[2], kde lze snadno vytvořit design. Výhodou použití této stránky je to, že je velmi snadno ovladatelná a hotové prvky lze snadno exportovat.

Serverovou část jsem psal na platformě .NET. Její výhodou je to, že hotovou aplikaci lze spustit na různých platformách operačního systému.

Poté jsem se pustil do psaní samotné hry pomocí Unity. Moje hra se skládá ze 3 scén (scenes).

4.1 Scéna připojení k serveru

Na první scéně je implementováno připojení na server. Na obrázku 1 zadáváme jméno uživatele a vybíráme místnost (lobby), ke které se připojíme.

Aby proběhlo připojení k serveru:

- Server musí být spuštěný.
- Pokud známe IP adresu serveru, můžeme použít pole pro zadání adresy a po stisknutí tlačítka „Try Connect“ se k němu připojíme a zobrazí se nám dostupné místnosti (lobby).
- Pokud neznáme adresu serveru, můžeme použít tlačítko „Refresh“ s obrázkem dvou šipek, které se otáčejí ve směru hodinových ručiček.
- Když se objeví místnost (lobby), musíme na ni kliknout.
- Když budeme mít vyplněné jméno uživatele a vybranou místnost (lobby), stiskneme tlačítko „Connect“.

Pokud jsme vše udělali správně, přejdeme na další scénu.

[2] Design hry

Obrázek 1: Scéna připojení k serveru

Na obrázku číslo 1 je zobrazeno:

- Pod číslem 1 – pole pro zadání uživatelského jména;
- Pod číslem 2 – tlačítko pro připojení k místnosti (lobby);
- Pod číslem 3 – tlačítko „Refresh“;
- Pod číslem 4.1 – pole pro zadání IP adresy serveru;
- Pod číslem 4.2 – tlačítko pro pokus o připojení podle uvedené IP adresy;
- Pod číslem 5 – prvek volné místnosti (lobby);
- Pod číslem 6 – tlačítko pro ukončení aplikace.

4.2 Scéna výběru herní figury

Na obrázku 2 je uživateli nabídnut výběr ze 4 herních figur: Mario, Princess, Yoshi a Donkey Kong. Po výběru figurky ji již nelze změnit. Kromě toho je k dispozici chat pro komunikaci mezi hráči. Po stisknutí tlačítka Ready se hra spustí pouze tehdy, pokud bude minimální počet hráčů 2.

Obrázek 2: Scéna výběru herní figury

Na obrázku číslo 2 je zobrazeno:

- Pod číslem 1 – výběr herní figury;
- Pod číslem 2 – tlačítko „Ready“;
- Pod číslem 3.1 – tlačítko pro odeslání zprávy;
- Pod číslem 3.2 – pole pro zprávu;
- Pod číslem 4 – tlačítko „Nastavení“, které otevírá malé okno s tlačítkem „Odpojit“.

4.3 Scéna hry

Na obrazovce číslo 3 se nachází rozhraní samotné hry. Zde probíhá samotný herní proces. Obrazovka je rozdělena na dvě části: herní pole a informace o hráči a průběhu hry.

V levé části obrazovky se nachází herní pole, se kterým hráč interaguje. Kliknutím na pole lze zobrazit informace o dané buňce: kdo je jejím vlastníkem, pokud se jedná o nemovitost, a kolik mincí se na ní nachází.

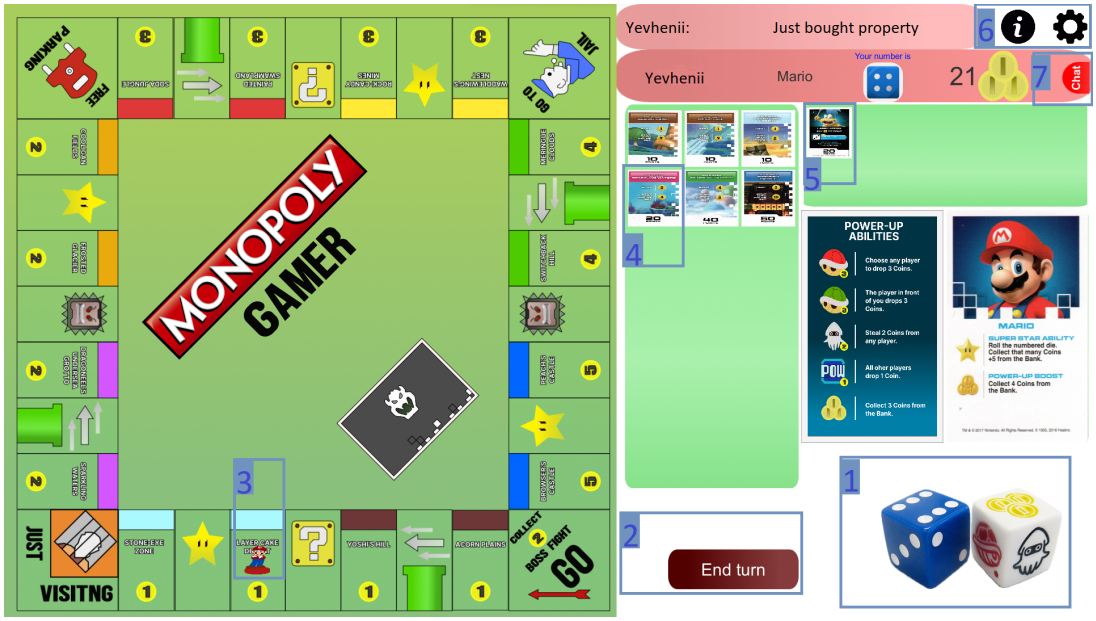
Kromě toho jsou zde zobrazena pomocná okna, jako například výběr první kostky nebo výběr hráče, na kterého chcete aplikovat různé akce.

V pravé části obrazovky se nacházejí kostky, pomocí kterých hráč začíná svůj tah, tlačítko pro ukončení tahu, tlačítko chatu, které otevře okno chatu. V horní části se nacházejí pole s informační řádkou, jménem hráče a jeho počtem mincí.

Na středu obrazovky se nacházejí kontejnery s informacemi o vlastnictví a poražených bossech. Karta hráče s jeho schopnostmi a karta s vlastnostmi každé schopnosti z krychle se super schopnostmi jsou také umístěny zde.

Když je na řadě náš tah, klikneme na tlačítko kostek a poté vybereme, kterou kostkou začít. Poté při tahu přesuneme hrací figurku na odpovídající počet políček tím, že přetáhneme hráčovu figurku na určené políčko. V závislosti na tom, na kterém políčku jsme skončili, se provádí určitá akce.

Při překročení políčka „Start“ začíná boj s bossem.



Obrázek 3: Scéna hlavní hry

Na obrázku číslo 3 je zobrazeno:

- Pod číslem 1 – hrací kostky – pokud je na řadě váš tah, klepněte na ně a v levé části obrazovky se zobrazí okno výběru, kterou kostku si vybrat první;
- Pod číslem 2 – tlačítko ukončení tahu „End turn“;
- Pod číslem 3 – Vybrána hrací figurka. Během tahu ji přetáhněte o počet políček odpovídající číslu na kostce;
- Pod číslem 4 – Karty nemovitostí, které vlastníte. Pokud na jednu z nich kliknete, zobrazí se v pravé části obrazovky její větší velikost;
- Pod číslem 5 – Karty bossů, které jste porazili. Pokud kliknete na jednu z nich, zobrazí se větší velikost v pravé části obrazovky;
- Pod číslem 6 – Tlačítka „Informace“ a „Nastavení“. Při kliknutí na tlačítko „Informace“ se otevře webová stránka s pravidly hry. Při kliknutí na tlačítko „Nastavení“ se otevírá malé okno s tlačítkem „Odpojit“;
- Pod číslem 7 – Tlačítko „Chatů“, při kliknutí otevírá malé okno s chatem.

5 Použití TCP a UDP protokolů

Nejprve bych chtěl říci, že třídy serveru a části klienta, která je zodpovědná za připojení, jsem vytvořil jako singleton[3].

Důvodem, proč jsem napsal část serveru a klienta ve formátu singletonu, je snadné sdílení jednoho objektu mezi různými částmi kódu a zajištění, že v programu existuje pouze jedna instance tohoto objektu. To může pomoct zabránit vzniku problémů s přístupem ke sdíleným prostředkům, jako jsou proměnné nebo soubory, a zlepšit celkovou správu paměti.

Navíc jsem zvolil číslo portu 8888. Toto číslo portu jsem vyhledal v tabulce, která je k dispozice na wikipedii.

5.1 Použití UDP protokolů

Použitím tohoto protokolu se klient snaží zjistit IP adresu serveru, ke kterému se připojuje. Server následně čeká na příchod balíčku (datagramu). Pokud dorazí balíček, server odešle klientovi svou IP adresu.

5.1.1 Implementace na straně klienta

Pro zjištění IP adresy serveru se používá asynchronní metoda. Výhodou použití asynchronní metody je v tom, že během svého života se nepoužívá hlavní vlákno aplikace. Téměř všechny metody, pomocí kterých klient komunikuje se serverem a naopak, jsou asynchronní.

Při psaní metody jsem počítal s tím, že nastavení lokální sítě je standardní a že IP adresy uživatelů jsou v rozsahu od 192.168.1.1 do 192.168.1.254. Metoda bere jako základ adresu uvedenou ve zdrojovém kódu 1 a pomocí cyklu prochází všechny možné adresy a odesílá datagramy.

```
1   // Set the IP address to ping
2   string ipAddress = "192.168.1." + i;
```

Zdrojový kód 1: C#

Viz zdrojový kód 2.

5.1.2 Implementace na straně serveru

Server čeká, až dorazí paket (datagram), jakmile ho dostane, odpoví klientovi a bude čekat, až se klient připojí pomocí TCP protokolu. Odpověď serveru obsahuje IP adresu. Viz zdrojový kód 3.

[3]Třída, ze které může existovat pouze jediná instance v průběhu životnosti aplikace a která poskytuje globální přístup k této instanci.

5.2 Použití TCP protokolu

Komunikace se provádí ve formátu JSON. Před tím, než každá strana něco odešle, musí být zpráva převedena do formátu JSON. To se provádí pomocí Serializeru a Deserializeru. K tomu byla vytvořena samostatná třída, která toto všechno zajišťuje.

5.2.1 Implementace na straně klienta

Když klient má určitou adresu serveru, snaží se k němu připojit. Pokud se pokus podaří, vytvoří se dva datové proudy, pro čtení a zápis. Poté se spustí hlavní metoda, ve které dochází k přijímání dat. Viz zdrojový kód 4.

5.2.2 Implementace na straně serveru

Rozhodl jsem se, že maximální počet hráčů, kteří se mohou připojit do hry, je 40 uživatelů. Výhodou TcpListener je to, že umožňuje omezit počet připojení, to se provádí během spuštění serveru.

Když se uživatel připojí, vytvoří se objekt, který ho zosobňuje. Tento objekt má metodu, která je zodpovědná za přijímání a odesílání paketů od klienta k serveru a naopak. Viz zdrojový kód 5.

6 Možné vylepšení v budoucnosti

Je několik možností, jak může být tato implementace vylepšena v budoucnosti. Například:

- Zabezpečení komunikace pomocí šifrování. Aktuálně jsou všechna data odesílána v otevřené podobě, což znamená, že jsou náchylná k útokům ze strany neoprávněných uživatelů. Šifrování by zajistilo, že data jsou chráněna a zabezpečena.

- Vylepšení použitelnosti uživatelského rozhraní. Aktuální implementace je poměrně základní a může být vylepšena pro lepší použitelnost pro uživatele. Například by mohla být vytvořena přehlednější a intuitivnější uživatelská rozhraní pro připojení k serveru a výběr herní místnosti.

- Implementace ukládání stavu hry. Aktuální implementace je schopna udržovat stav hry pouze po dobu běhu serveru. Pokud se server vypne nebo restartuje, všechny uložené informace jsou ztraceny. Implementace ukládání stavu hry by umožnila uživatelům obnovit hru ze stavu, ve kterém byla přerušena, například v případě výpadku spojení nebo odpojení uživatele.

- Optimalizace serverového kódu. Pokud by se hra rozrostla a obsahovala víc hráčů, mohlo by být nutné optimalizovat serverový kód, aby byl schopen zvládnout větší zátěž. To může zahrnovat optimalizaci kódu, optimalizaci použití paměti a škálování serveru.

- Možné vylepšení hry by mohlo spočívat v přidání hracích figurek a karet bosů. Čím více herních figurek bude k dispozici, tím bude hra zajímavější a různorodější.

- Možným vylepšením hry, která je momentálně omezená pouze na hraní v lokální síti, by mohlo být umožnění připojení hráčů ze vzdálených sítí. Pro dosažení tohoto cíle by bylo třeba implementovat další vrstvu komunikace a přizpůsobit ji potřebám vzdáleného připojení. To by zajistilo větší flexibilitu a dostupnost hry pro hráče, kteří se nenacházejí v blízkosti fyzického umístění serveru.

```
1  private async Task<IPAddress> FindServer()
2  {
3    // Create a UDP client to send the ping requests
4    UdpClient udpClient = new UdpClient();
5
6    byte[] message = Encoding.ASCII.GetBytes("ARE YOU MONOSERVER");
7
8    // Loop through all possible IP addresses on the local network
9    for (int i = 1; i < 255; i++)
10   {
11     // Set the IP address to ping
12     string ipAddress = "192.168.1." + i;
13
14     // Send the ping request to the IP address
15     await udpClient.SendAsync(message, message.Length, new IPEndPoint
           (IPAddress.Parse(ipAddress), 8888));
16
17     // Wait for a response from the server for 1 second
18     if (udpClient.Client.Poll(1000, SelectMode.SelectRead))
19     {
20       // Receive the response from the server
21       IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);
22       var ans = (await udpClient.ReceiveAsync());
23       var response = ans.Buffer;
24       remoteEP = ans.RemoteEndPoint;
25       if (Encoding.ASCII.GetString(response).Contains("YES"))
26       {
27         // Print the server IP address
28         Console.WriteLine("Server IP: " + remoteEP.Address);
29         udpClient.Close();
30         var res = Encoding.ASCII.GetString(response);
31         return IPAddress.Parse(res.Substring(4));
32       }
33     }
34     // Wait for a short time before sending the next ping request
35     Thread.Sleep(100);
36   }
37
38   // Close the UDP client
39   udpClient.Close();
40   return null;
41 }
```

Zdrojový kód 2: C#

```
1  private async Task UDPListenerForTCpConnection()
2  {
3    while (true)
4    {
5      try
6      {
7        // Receive the ping request from the client
8        IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);
9        byte[] request = udpListener.Receive(ref remoteEP);
10
11       // Check if the request is a ping request
12       if (Encoding.ASCII.GetString(request) == "ARE YOU MONOSERVER")
13       {
14         Console.WriteLine("Get Packet from " + remoteEP.Address);
15         // Send the server IP address to the client
16         byte[] response = Encoding.ASCII.GetBytes("YES " +
               GetLocalIPAddress());
17         udpListener.Send(response, response.Length, remoteEP);
18         CreateConnection();
19       }
20     }
21     catch (Exception e)
22     {
23       Console.WriteLine(e.Message);
24     }
25   }
26 }
```

Zdrojový kód 3: C#

```
1  public async Task ConnectToServer(bool ownServer = false, IPAddress
       ownServerIp = null)
2  {
3    if (!ownServer && ownServerIp == null)
4    {
5      this.ip = await FindServer();
6      if (this.ip == null)
7      return;
8    }
9    else
10   {
11     this.ip = ownServerIp;
12   }
13
14   tcpClient = new TcpClient();
15
16   try
17   {
18     tcpClient.Connect(this.ip, this.port);
19     reader = new StreamReader(tcpClient.GetStream());
20     writer = new StreamWriter(tcpClient.GetStream());
21     if (writer is null || reader is null) return;
22     Connected = tcpClient.Client.Connected;
23     Task.Run(async () => await ReceiveMessageAsync(reader));
24   }
25   catch (Exception ex)
26   {
27     Connected = false;
28     Debug.Log(ex.Message);
29   }
30 }
```

Zdrojový kód 4: C#

```
async Task createConnectionLoop()
{
  int i = 0;
  while (i < 40)
  {
    i++;
    Task.Run(() => CreateConnection());
  }
}
private async void CreateConnection()
{
  try
  {
    var tcpClient = tcpListener.AcceptTcpClient();
    TCPClientObj clientObj = new TCPClientObj(tcpClient, this);
    Console.WriteLine("Client has been connected from {0} with ID:
        {1}", tcpClient.Client.RemoteEndPoint, clientObj.ID);
    this.clients.Add(clientObj);
    Task.Run(() => clientObj.ProcessAsync());
  }
  catch (Exception ex) {
    foreach(var cl in clients)
    {
      if (!cl._client.Connected)
      {
        cl.Close();
        cl._client.Close();
      }
    }
    CreateConnection();
  }

}
public async Task StartServerAsync()
{
  Console.WriteLine("Starting server....");
  Console.WriteLine(GetLocalIPAddress());
  try
  {
    tcpListener.Start(40);
    Task.Run(async () => await UDPListenerForTCpConnection());
    Task.Run(async () => await createConnectionLoop());
  }
  catch (Exception ex)
  {
    Console.WriteLine(ex.Message);
  }
}
```

Zdrojový kód 5: C#

Závěr

V rámci bakalářské práce byly implementovány serverová a klientská část hry. Díky použitým technologiím je možné samotnou hru i server spustit na různých platformách.

Jelikož hru nelze šířit, je možné ji hrát pouze v lokální síti, vše pro to bylo však uděláno.

Při psaní práce jsem získal dovednosti práce s TCP a UDP protokoly s použitím programovacího jazyka C#. Kromě toho jsem získal nové zkušenosti s prací v Unity.

Použil jsem standardní knihovny System.Net a System.Net.Sockets pro vytvoření TCP serveru a klienta. Server používá třídu TcpListener a klient používá TcpClient. Předtím, než jsem zvolil TcpListener a TcpClient, mohl jsem také zvolit Socket, ale tento způsob jsem zamítl, protože Socket se většinou používá k přenosu souborů různých velikostí, mediálních souborů atd. V komunikaci klient-server používám pouze textový formát (string). Proto není potřeba přenášet soubory, pouze textový formát. TcpListener/TcpClient jsou vytvořeny na základě Socket.

Pro rozpoznávání odeslaných dat jsem použil JSON serializer/deserializer, které jsem napsal sám, a které využívají základní knihovnu System.Text.Json.

Když odesílám data používám asynchronní metodu .WriteLineAsync a v případě, že dojde k několika událostem, které také potřebují něco odeslat, existuje pravděpodobnost, že data budou zapsána současně a server nebude schopen rozpoznat odeslané informace. Aby se to nestalo, používám asynchronní SemaphoreSlim, aby několik vláken nemohlo zapisovat současně.

Conclusions

Within the bachelor's thesis, the server and client parts of the game were implemented. Thanks to the used technologies used, it is possible to run both the game and the server on various platforms.

Since the game cannot be distributed, it can only be played on a local network, but everything has been done for that purpose.

While writing the thesis, I gained skills in working with TCP and UDP protocols using the C# programming language. In addition, I gained new experience in working with Unity.

I used the standard libraries System.Net and System.Net.Sockets to create a TCP server and client. The server uses the TcpListener class and the client uses the TcpClient class. Before choosing TcpListener and TcpClient, I could have also chosen Socket, but I rejected this approach because Socket is mostly used for transferring files of different sizes, media files, etc. In client-server communication, I only use the text format (string). Therefore, there is no need to transfer files, only the text format. TcpListener/TcpClient are created based on Socket.

To recognize the sent data, I used a JSON serializer/deserializer, which I wrote myself and which uses the basic System.Text.Json library.

When sending data, I use the asynchronous method .WriteLineAsync and in case several events occur that also need to send something, there is a possibility that the data will be written simultaneously and the server will not be able to recognize the sent information. To prevent this, I use an asynchronous SemaphoreSlim so that several threads cannot write at the same time.

A Obsah elektronických dat

`doc/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

`src/`
: Kompletní zdrojové kódy aplikace MonopolyGamer a MonopolyGamerServer. Ve složkách s příslušnými jmény budou uloženy zdrojové kódy aplikace.

`README.txt`
: Kompletní instrukce pro instalaci a spuštění aplikace MonopolyGamer a MonopolyGamerServer.

Bibliografie

[1] JSON Dostupný z: [https://www.json.org/json-en.html]

[2] Unity Dostupný z: [https://docs.unity.com/]

[3] Libor Dostálek, Alena Kabelová, Velký průvodce protokoly TCP/IP a systémem DNS. 5. aktualizované vydání, Computer Press, a.s., 2008, ISBN: 978-80-251-2236-5

[4] Richard Blum, C# Network Programming, Sybex, 2003, ISBN: 0782141765