Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Online varianta hry Blokus

2017

Tomáš Vlk

Vedoucí práce: Mgr. Petr Krajča, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Tomáš Vlk
Název práce:	Online varianta hry Blokus
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	49
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Tomáš Vlk
Title:	Blokus-like online game
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	49
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem bakalářské práce bylo vytvořit síťovou variantu hry Blokus s počítačovým hráčem, spustitelnou na více platformách. Výsledná aplikace je implementována pro dva, nebo čtyři hráče s možností hry po síti a je spustitelná na platformách Windows, Linux, macOS a Android.

Synopsis

The main goal of the bachelor thesis was to create a network variant of the Blokus game with a computer player and executable on multiple platforms. The final application is implemented for two or four players with network playability and it is executable on Windows, Linux, macOS and Android.

Klíčová slova: hra; hra po síti; počítačový hráč; klient; server; Java

Keywords: game; online game; bot; client; server; Java

Děkuji panu doktorovi Petru Krajčovi za vedení bakalářské práce, zvláště za užitečné rady a podněty při konzultacích.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

1.1 Zadání bakalářské práce

Cílem práce je vytvořit online variantu hry Blokus. Výsledná aplikace by měla podporovat hru více hráčů, počítačového hráče a mít nativního klienta pro více platforem.

1.2 Motivace k tvorbě hry

Důvodů, proč jsem si pro svoji bakalářskou práci vybral právě tvorbu hry, je několik. Mezi hlavní důvody patří ten, že od malička jsem velmi soutěživý a mám rád jakékoliv hry, ať už se jedná o hry sportovní, počítačové a nebo právě deskové. Nutno ale dodat, že konkrétně o hře Blokus jsem se poprvé dozvěděl právě tehdy, když jsem přemýšlel nad tématem své práce. Hra mě na první pohled zaujala zejména tím, jak dokáže být každá partie odlišná.

1.3 Blokus

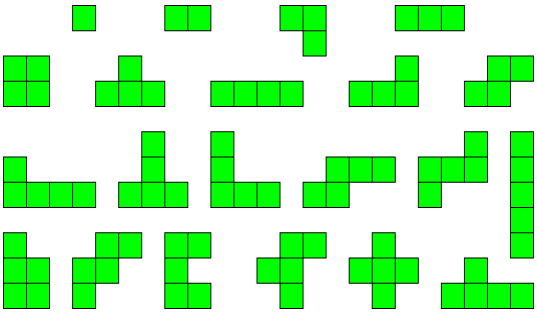
Pro popis hry Blokus bylo čerpáno ze zdroje informací [6]. Blokus je desková hra pro 2 až 4 hráče, založená na postupném pokládání kostiček na hrací desku o rozměrech 20x20 políček. Autorem hry je Bernard Tavitian. Blokus vydala poprvé v roce 2000 firma Sekkoia.

1.3.1 Pravidla hry

Na začátku hry má každý hráč k dispozici 21 různých kostiček jedné barvy (modrá, žlutá, červená a zelená). Kostičky jsou tvořeny z jednoho až pěti dílků, viz obrázek 1. Hráči se postupně střídají v tazích podle barvy svých kostiček. Jako první táhne modrý, po něm žlutý, červený a nakonec zelený. Hráči postupně pokládají na hrací desku svoje kostičky podle následujících pravidel:

- První tah každého hráče musí být v některém z rohů hrací desky.
- Položená kostička nesmí překrývat jinou.
- Kostičky stejné barvy se mohou dotýkat pouze rohem.
- Kostičky stejné barvy se musí dotýkat alespoň jedním rohem.
- Kostičky různých barev se mohou dotýkat libovolně.
- Pokud některý z hráčů nemůže položit žádnou kostičku, je daný hráč vyřazen a hra pokračuje.
- Hra končí, pokud nikdo nemůže položit žádnou kostičku.

- Bodové ohodnocení je založeno na počtu jednotlivých dílků, ze kterých se kostičky skládají, například kostička v levém dolním rohu na obrázku 1 má hodnotu 5 bodů.
- Pokud hráč položil všechny kostičky, a navíc jako poslední položil kostičku tvořenou jedním dílkem, získá bonus 20 bodů. Pokud položil jako poslední kostičku jinou, získá bonus 15 bodů.
- Vítězem je hráč s nejvíce body.



Obrázek 1: Sada všech kostiček

1.3.2 Dostupné varianty

Aplikace podporuje dva režimy hry, buď může uživatel hrát pouze proti počítači, nebo může hrát přes server s ostatními hráči. Hra přes server vyžaduje internetové připojení, naopak hra pouze proti počítači nikoliv. Obě varianty jsou implementovány pro 2, nebo 4 hráče.

Pro 2 hráče
Hra dvou hráčů, kde každý z nich má k dispozici sadu kostiček jedné barvy, je dostupná ve dvou variantách:

- Na začátku je hrací deska prázdná.
- Před začátkem hry se na hrací desku náhodně umístí celá sada kostiček červené barvy, které následně slouží pro hráče jako překážky.

Pro 4 hráče
Každý z hráčů dostane k dispozici sadu kostiček jedné barvy a postupně se střídají v tazích, dokud hra neskončí.

1.3.3 Ostatní varianty

Níže popsané varianty nejsou v aplikaci implementovány, ale mohou být použity jako budoucí rozšíření.

Další varianta hry pro 2 hráče
Každý hráč dostane na začátku hry dvě sady kostiček (například červené a zelené). Hráč začíná ve dvou rozích a kostičky dvou barev stejného hráče se mohou dotýkat libovolně.

Pro 3 hráče
Hráči se postupně střídají v pokládání kostiček čtvrté barvy, ale její konečné skóre se nepočítá. Varianta pro 3 hráče bývá považována za nespravedlivou, protože v ní má výhodu ten hráč, který začínal v rohu naproti čtvrté barvě.

Hra v týmech
Čtyři hráči hrají ve dvojicích proti sobě. První dvojice má k dispozici kostičky modré a červené barvy, druhá dvojice žluté a zelené. Pořadí tahů je i nadále určeno barvou kostiček, hráči obou dvojic se tedy v tazích střídají.

2 Rozdělení práce

Jelikož cílem práce je vytvořit aplikaci pro více platforem, navíc podporující hru po síti a počítačového hráče, nabízí se výslednou práci rozdělit do následujících částí:

- **Počítačový klient** – Spustitelný na platformách Windows, Linux a macOS. U implementace počítačového klienta bylo využito zdroje informací [1].
- **Klient pro Android** – U implementace klienta pro Android bylo využito zdroje informací [4] a [5].
- **Server** – U implemetace serveru bylo využito zdroje informací [2].

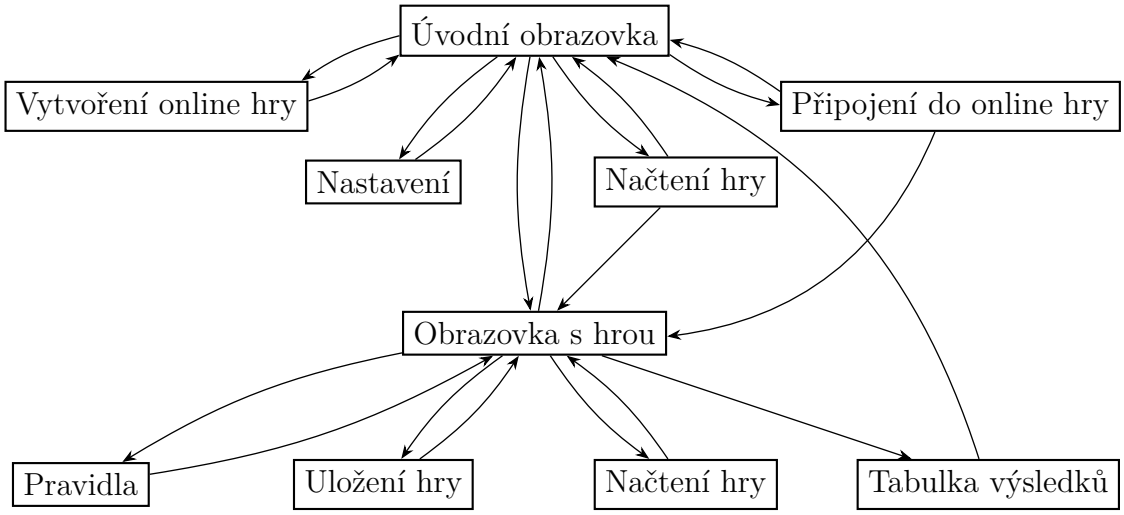
Počítačový klient a klient pro Android zprostředkovávají hráčům uživatelské rozhraní hry a server slouží jako prostředník pro komunikaci mezi jednotlivými klienty při hře po síti.

Pokud uživatel zvolí hru pouze proti počítači, má výpočet dalšího tahu počítačového hráče na starost daný klient. Při hře po síti zajišťuje výpočet server.

3 Klient

V této kapitole je shrnuta funkčnost počítačového klienta a klienta pro Android. Jejich účel je totiž stejný.

Počítačový klient se skládá z několika obrazovek (oken). Na počítačovém klientu mají tyto obrazovky fixní velikost. To zejména z toho důvodu, aby byly zachovány přívětivé rozměry jednotlivých grafických komponent v dané obrazovce. Pokud uvažujeme klienta pro Android, tak se těmto obrazovkám říká aktivity. Hierarchie těchto obrazovek, respektive aktivit, je stejná pro oba klienty a je ilustrována na obrázku 2. Každá z těchto obrazovek má vlastní funkcionalitu, ta se může lišit, pokud uživatel hraje po síti, nebo pouze proti počítači.



Obrázek 2: Hierarchie obrazovek

3.1 Úvodní obrazovka

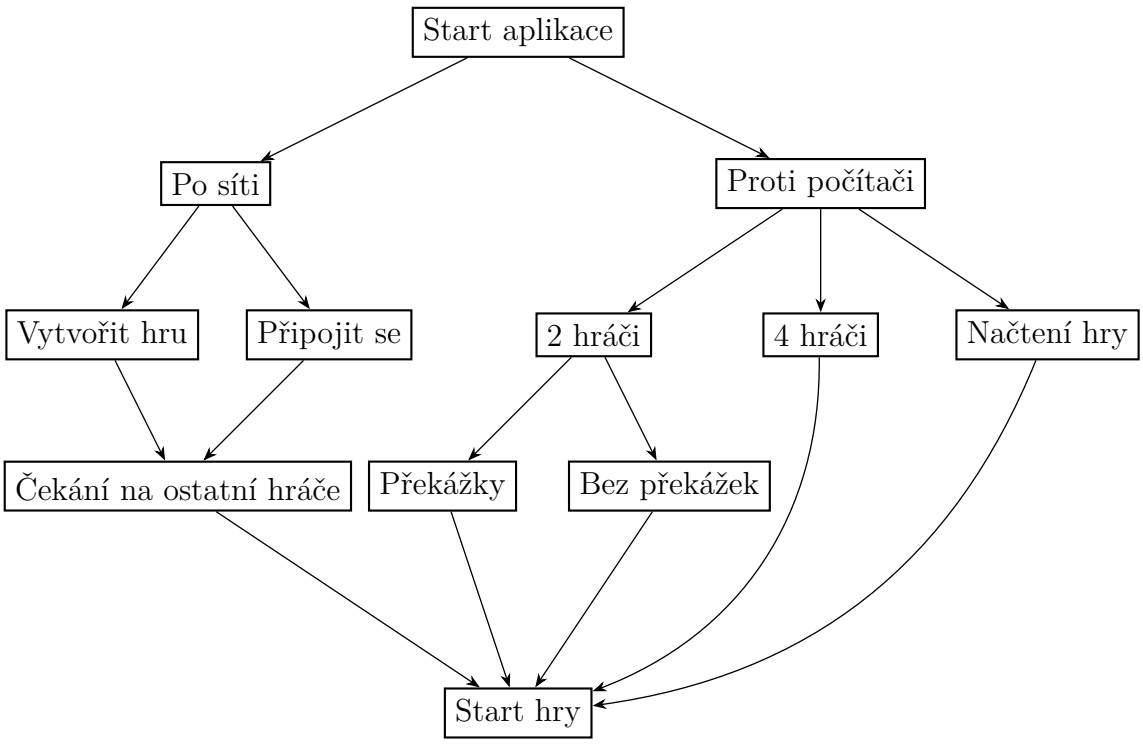
Po spuštění aplikace představující klienta se uživateli zobrazí úvodní obrazovka, viz obrázek 3. Účelem úvodní obrazovky je dostat uživatele do samotné hry. To lze provést několika způsoby, jež závisí na tom, zda si uživatel přeje hrát po síti, nebo pouze proti počítači. Pokud uživatel chce hrát po síti, má dvě možnosti, buď se může připojit do již existující hry, nebo vytvořit hru novou. Při hře pouze proti počítači si může hráč vybrat, zda chce hru pro dva, nebo čtyři hráče. Pokud zvolí hru pro dva, může si dále vybrat, jestli chce na hrací desce vygenerovat překážky. Vytváření hry proti počítači ovlivní následující faktory:

- Celkový počet hráčů.
- Překážky (pouze pro 2 hráče).

Diagram všech možností, jak se lze z úvodní obrazovky dostat do samotné hry, je uveden na obrázku 4.

Rozdílný start hry
Nutno zmínit, že pokud uživatel hraje po síti, tak se vstup do samotné hry může opozdit. To je způsobeno čekáním na připojení ostatních hráčů. Naproti tomu vstup do hry proti počítači je okamžitý.

Obrázek 3: Úvodní obrazovka



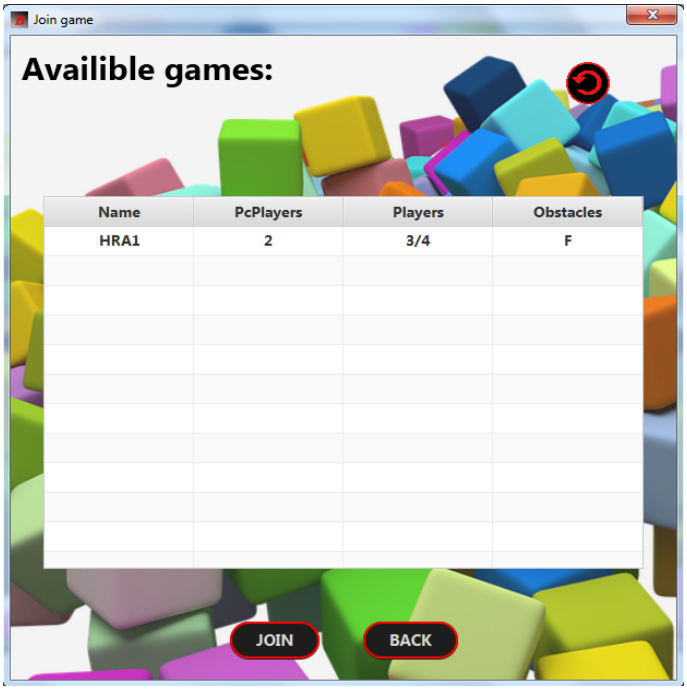
Obrázek 4: Diagram možností, jak vstoupit do hry

3.2 Obrazovka pro připojení do hry po síti

Tato obrazovka zobrazuje uživateli již vytvořené hry po síti v jednoduché tabulce, jak je ilustrováno na obrázku 5. V tabulce jsou uvedeny pouze takové hry, do kterých ještě není připojen požadovaný počet hráčů. Tedy probíhající hry se zde nezobrazují. Z této tabulky může uživatel také vyčíst základní informace o hře, konkrétně:

- Jméno hry.
- Celkový počet hráčů.
- Počet již přihlášených hráčů.
- Počet počítačových hráčů.
- Zda se jedná o hru s překážkami.

Pokud si zde uživatel vybere hru, do které se chce připojit, mohou nastat dvě možnosti. Buď je ihned zobrazena obrazovka s hrou, to znamená, že daný hráč byl poslední na kterého se čekalo, nebo je zahájeno čekání na ostatní hráče. Z této obrazovky se lze vrátit na úvodní obrazovku i bez připojení do hry.



Obrázek 5: Obrazovka pro připojení do hry po síti

3.3 Obrazovka pro vytvoření hry po síti

Obrazovka pro vytvoření hry po síti představuje jednoduchý dialog, viz obrázek 6. Tento dialog umožní hráči navolit si, s jakými parametry chce vytvořit novou hru po síti. Tyto parametry jsou:

- Jméno hry.
- Celkový počet hráčů.
- Počet počítačových hráčů.
- Překážky (pouze pro 2 hráče).

Parametr jméno hry dále slouží jako jednoznačný identifikátor, přes který se ostatní hráči mohou do této hry připojit. Není tedy možné vytvořit dvě hry se stejným jménem. Pokud je hra úspěšně vytvořena, je zahájeno čekání na ostatní hráče. Uživatel může toto čekání ukončit, například pokud se mu zdá, že trvá příliš dlouho. V případě, že uživatel již nechce vytvořit novou hru, lze se z této obrazovky vrátit na úvodní obrazovku bez toho, aniž by byla hra vytvořena.

Obrázek 6: Obrazovka pro vytvoření hry po síti

3.4 Obrazovka s nastavením

Poté, co uživatel vstoupí do obrazovky s nastavením, má možnost změnit IP adresu serveru, přes který chce hrát. Tato změna se projeví pouze, pokud bude uživatel hrát po síti.

3.5 Obrazovka s hrou

Obrazovka s hrou je jednou z nejdůležitějších obrazovek celé aplikace, na obrázku 7 je ilustrována obrazovka s hrou pro Android klienta, na obrázku 8 pro počítačového klienta. Tato obrazovka poskytuje hráčům uživatelské rozhraní pro hru Blokus a jejími základními prvky jsou:

- **Hrací kostičky** – obrázky všech kostiček, se kterými může hráč ještě provést tah.
- **Hrací deska** – plocha tvořená čtyřmi sty políčky, na kterou hráči postupně pokládají své kostičky.
- **Ukazatel skóre** – tabulka v pravém horním rohu obrazovky zobrazující aktuální počet bodů, získaný jednotlivými hráči.
- **Hráčem vybraná kostička** – kostička, kterou si hráč vybral pro svůj další tah. Před tím, než hráč tuto kostičku umístí na hrací desku, může jí libovolně otáčet či zrcadlově obracet.
- **Menu** – navigační panel, kde jsou dostupné následující akce:
  - Zobrazit pravidla hry.
  - Načíst hru.
  - Uložit hru.
  - Vzdát hru.
  - Návrat do hlavního menu (úvodní obrazovka).
  - Ukončení celé aplikace.

3.5.1 Položení kostičky na hrací desku

Co se týče ovládání samotné obrazovky s hrou, je nutno uvést, jak položit kostičku na hrací desku. Tato akce se u počítačového klienta a klienta pro Android **liší**. Nicméně při pokládání mohou nastat dvě situace:

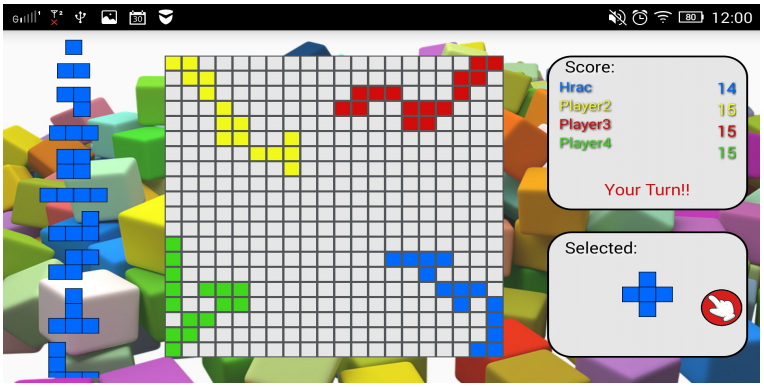
- Pozice kostičky **vyhovuje** pravidlům – tedy je umístěna na desku a hra pokračuje.
- Pozice kostičky **nevyhovuje** pravidlům – hráč je informován o špatné pozici jeho kostičky a musí opakovat svůj tah.

Položení kostičky na počítačovém klientu
Nejdříve si hráč zvolí libovolnou z ještě dostupných kostiček, ty jsou vyobrazeny v pravém dolním rohu. Zvolenou kostičku může hráč transformovat podle svého uvážení dvěma způsoby. Buď pomocí dostupných tlačítek, nebo pokud posune myš nad hrací desku, bude zpřístupněna transformace kostičky pomocí pravého tlačítka myši. Samotné položení kostičky provede hráč posunutím myši nad hrací

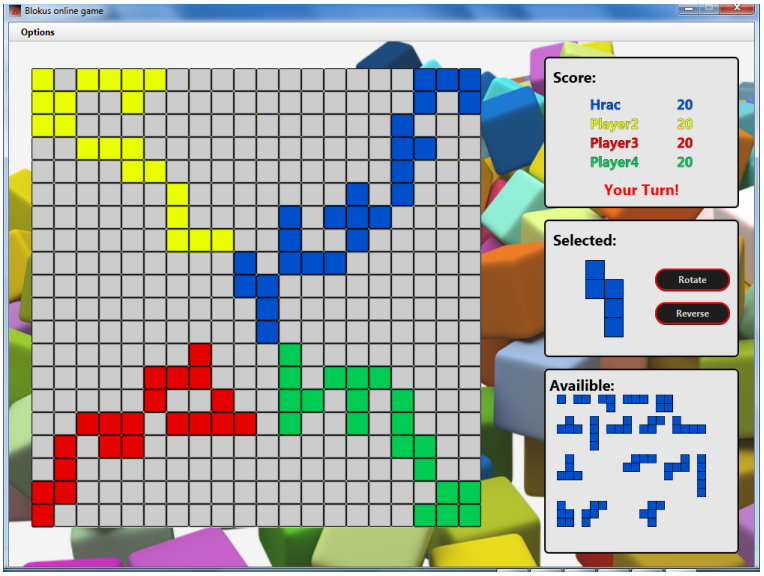
desku, a až bude spokojený s pozicí, kde se mu kostička vykreslí, stačí stisknout levé tlačítko myši.

Položení kostičky na Androidu

Stejně jako na počítačovém klientu si hráč nejprve vybere jednu z ještě dostupných kostiček. Dostupné kostičky jsou zde zobrazeny v posuvné obrazovce na levé části displeje. Vybranou kostičku lze transformovat jednoduchým kliknutím na ni. Umístění kostičky na hrací desku je realizováno pomocí Drag&Drop[1]. Stačí tedy kliknout na ikonku vyobrazenou vedle vybrané kostičky, tu pak tahem prstu přemístit na požadovanou pozici a uvolnit stisk.



Obrázek 7: Obrazovka s hrou u klienta pro Android

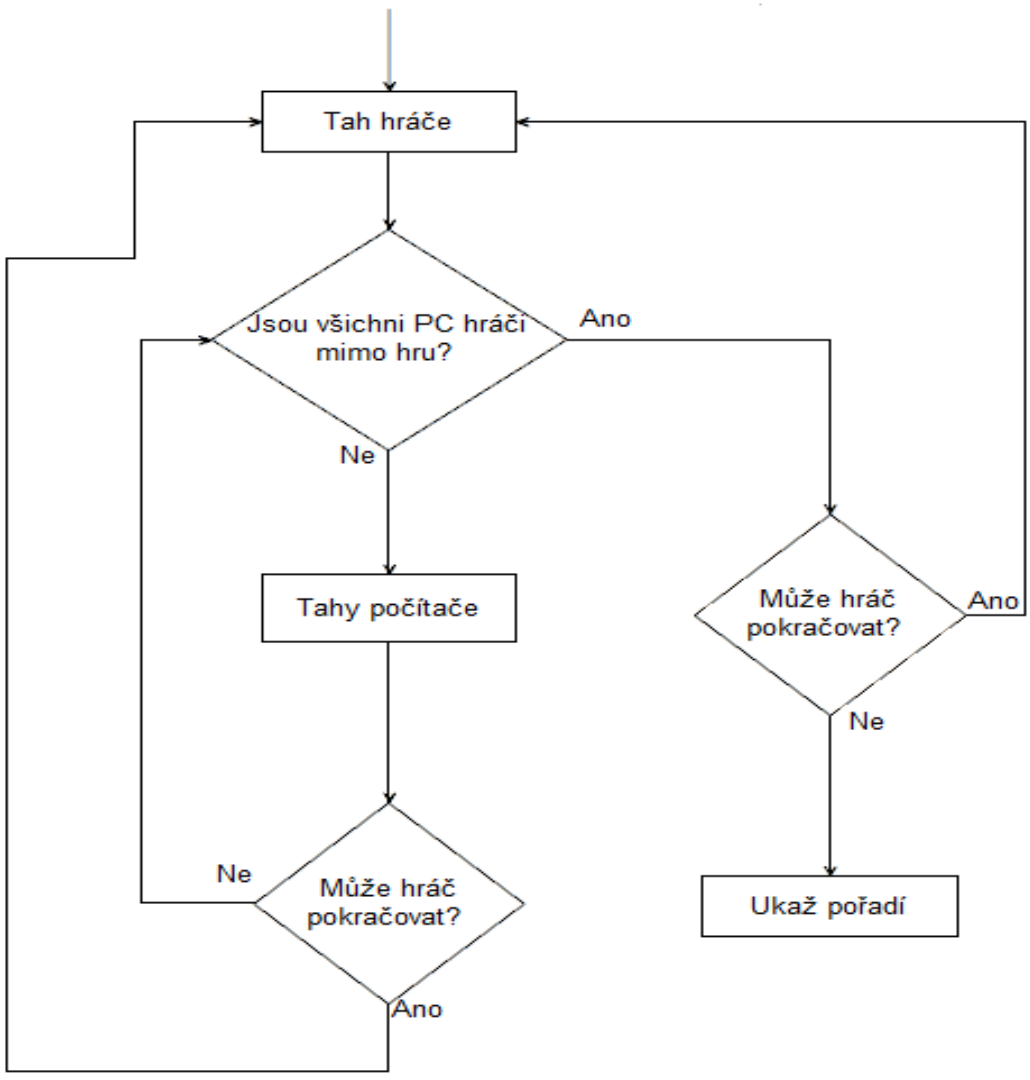


Obrázek 8: Obrazovka s hrou u počítačového klienta

[1]Technika sloužící k přemisťování objektů na obrazovce pomocí pohybu prstu.

3.5.2 Průběh hry proti počítači

Při hře proti počítači má první tah vždy člověk, poté následují tahy počítačových hráčů. Tímto způsobem se tahy střídají, dokud alespoň jeden hráč má možnost, jak pokračovat. Pokud jsou všichni hráči mimo hru, tak se ověří, zda má některý z hráčů nárok na bonusové body, viz sekce 1.3.1. Po kontrole skóre se hráčům zobrazí tabulka s pořadím. Průběh hry proti počítači je ilustrován na obrázku 9. Způsob výpočtu tahu počítačového hráče je popsán v kapitole 7.



Obrázek 9: Průběh hry proti počítači

3.5.3 Průběh hry po síti

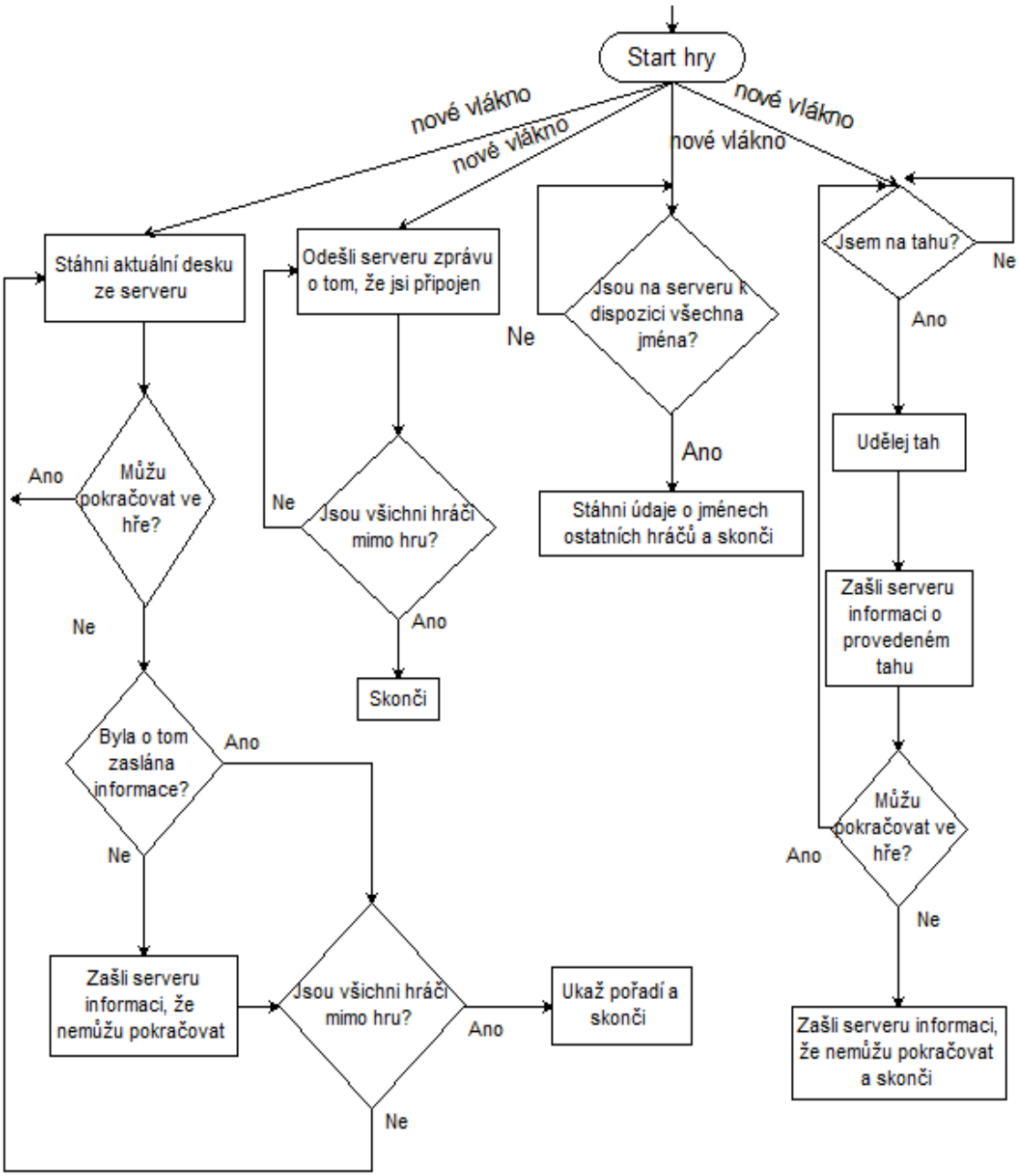
Hra po síti se od hry proti počítači liší v jedné zásadní věci. Všichni klienti synchronizují důležitá data se serverem, viz kapitola 5. Komunikace probíhá v periodických časových intervalech a využívá k tomu čtyři vlákna, jejich účel a popis je uveden v sekci 4.4.

Rozdělení průběhu hry po síti
Průběh hry po síti je rozdělen na tři logické části. Toto rozdělení není ve zdrojovém kódu nijak implementováno a je zde uvedeno pouze za účelem přehlednosti. Diagram průběhu hry po síti je znázorněn na obrázku 10.

- **Inicializace** – nastává po přepnutí z úvodní obrazovky do obrazovky s hrou. Klient v této fázi odešle na server jméno daného hráče. Dále spustí všechna vlákna komunikující se serverem.

- **Samotná hra** (Od prvního do posledního tahu) – na základě dat získaných od serveru je daný hráč vyzván, aby provedl tah. Pokud hráč není na tahu, tak pouze sleduje tahy ostatních hráčů. Nastane-li situace, kdy hráč už nemá možnost, jak pokračovat dál ve hře, informuje o tom server a nadále je pouze divákem.

- **Konec hry** (Pokud žádný hráč nemá možnost jak pokračovat) – v této situaci jsou zastavena všechna zbývající vlákna, která komunikují se serverem. Proběhne případné přičtení bonusů jednotlivým hráčům. Klient se následně odpojí od hry a přejde na obrazovku s pořadím, viz 3.9.



Obrázek 10: Průběh hry po síti

3.6 Obrazovka pro uložení hry

Tato obrazovka slouží pro uložení hry. Uživatel je zde vyzván, aby vložil název, pod kterým chce danou hru uložit. Následně jsou do XML souboru zapsána všechna data, která jsou potřebná k pozdějšímu obnovení hry. Soubor s uloženou hrou obsahuje informace o:

- Jménu hráče.
- Počtu hráčů.
- Stavu hrací desky.
- Kostičkách, které hráči už nemohou použít – při obnovení hry jsou ze všech kostiček přidělených danému hráči odebrány právě tyto kostičky.
- Počtu tahů počítačových hráčů – důležité pro první tahy počítačového hráče, viz kapitola 7.

Ukládání síťových her v aplikaci není implementováno, jelikož hráči, kteří spolu hrají, se nemusí navzájem znát. Může pro ně být tedy značně složité se domluvit, kdy by chtěli uloženou hru dohrát. Ukládání her po síti by také vyžadovalo značnou režii serveru.

3.7 Obrazovka pro načtení hry

Na této obrazovce se uživateli zobrazí tabulka jmen uložených her. Ten může jednotlivé hry buď načíst a pokračovat tak v uložené hře, nebo může hru odstranit. Pokud uživatel chce hru načíst, jsou z XML souboru odpovídajícímu dané hře načtena všechna důležitá data, popsána v sekci 3.6. Pomocí těchto dat je poté aktualizován stav obrazovky s hrou a ta může pokračovat. Pokud chce uživatel hru odstranit, je smazán soubor odpovídající dané hře, následně je aktualizováno grafické rozhraní, kde smazaná hra již nefiguruje.

3.8 Obrazovka s pravidly

Do obrazovky s pravidly se uživatel dostane přes hlavní menu na obrazovce s hrou, jsou zde popsána pravidla hry, tak jak je uvedeno v sekci 1.3.1.

3.9 Obrazovka s pořadím

Jak už název napovídá, tato obrazovka slouží k vyobrazení celkového pořadí. Do této obrazovky se nedá dostat pomocí žádného tlačítka nebo klávesové zkratky. Klient automaticky přepne na obrazovku s pořadím, jakmile rozpozná konec hry. Z této obrazovky se lze přepnout zpět do hlavního menu, nebo celou aplikaci ukončit.

4 Implementace klienta

4.1 Hra

Při vstupu do hry dostane každý hráč své ID. Při hře dvou hráčů je z intervalu $[1, 2]$, při hře čtyř hráčů z intervalu $[1, 4]$. Samotnou hru představuje obrazovka s hrou. Důležité třídy pro reprezentaci jednotlivých prvků této obrazovky jsou:

- `GameBoard` – třída reprezentující hrací desku. Ta zde figuruje jako dvojrozměrné pole typu `integer` o velikosti 20x20. Obsahuje metody sloužící k operacím nad danou hrou, jako například zapsaní tahu daného hráče na desku nebo ověření toho, zda může daný hráč ještě pokračovat ve hře. Grafickou podobu představuje čtyři sta tlačítek, kde každé tlačítko má jednoznačné ID $\in [0, 399]$, které přesně odpovídá pozici v dvourozměrném poli.

- `Brick` – představuje jednu konkrétní kostičku. Barvu kostičky určuje ID daného hráče. Každá kostička je jednoznačně definována body v dvourozměrném prostoru s tím, že souřadnice počátku určuje levý horní roh. Například kostička v levém dolním rohu obrázku 1 je definována body:

  (0,0),(0,-1),(0,-2),(1,-1),(1,-2)

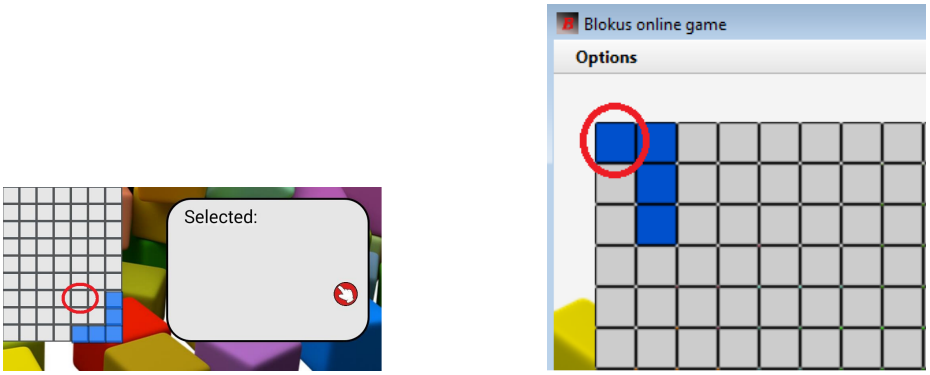
- `GameScreenController` – třída obsahující veškeré metody pracující s uživatelským rozhraním obrazovky s hrou, jako jsou metody pro položení kostičky, uložení hry, načtení hry apod. V případě hry po síti je zde realizována periodická komunikace se serverem, viz 4.4. Stejnou úlohu jako třída `GameScreenController` plní na platformě Android třída `GameActivity`.

4.2 Položení kostičky

Při pokládání kostičky na hrací desku jde v zásadě o zapsaní údajů do dvourozměrného pole třídy `GameBoard` a následné obarvení odpovídajících tlačítek na obrazovce s hrou. Na souřadnicích *(x,y)* v poli, které reprezentuje hrací desku, mohou být dvě hodnoty:

- **0** – Žádný hráč nepoložil kostičku na tyto souřadnice.
- **ID** $\in [1, 4]$ – Hráč s daným ID položil kostičku na tyto souřadnice.

Při pokládání kostičky je rozdíl mezi počítačovým klientem a Androidem v tom, že v prvním případě je metoda obsluhující položení kostičky navázána na událost kliknutí, zatímco u Androidu je navázána na událost uvolnění stisku (tzv. Drop). Nicméně v obou případech je metoda zavolána s ID tlačítka, nad kterým byl levý horní dílek vybrané kostičky, jak je znázorněno na obrázku 11.



Obrázek 11: Ukázky pokládání kostičky (za levý horní roh)

Pokud známe ID tlačítka, se kterým byla zavolána metoda obsluhující položení kostičky a souřadnice bodů definujících kostičku, lze přesně určit pozici na hrací desce, kam má být kostička umístěna. Než se tak provede, je ověřeno, že daná pozice vyhovuje pravidlům hry tak, jak jsou popsána v sekci 1.3.1. Pokud pozice vyhovuje všem pravidlům, jsou na příslušné souřadnice v dvourozměrném poli třídy `GameBoard` zapsána ID hráče, který kostičku položil. Na základě těchto změň se aktualizuje grafické rozhraní (obarví se tlačítka) a hra pokračuje.

U klientu pro Android je při pokládání kostičky nejdříve odstraněn obrázek kostičky, kterým uživatel pohybuje, a následně je ověřeno, zda pozice vyhovuje pravidlům. Poté jsou vybarvena příslušná políčka na hrací desce. Tím je způsobeno, že kostička zmizí z obrazovky a po menší časové prodlevě jsou vybarvena políčka. Časovou prodlevu způsobuje doba trvání ověřovacího výpočtu.

4.3 Volba IP adresy serveru

Předpokládá se, že aplikace realizující server bude spuštěna na zařízení, které je dostupné přes veřejnou IP adresu. Právě na tuto adresu se potom budou všichni klienti připojovat, pokud uživatel neřekne jinak. Protože aplikace realizující server zatím na žádném takovém zařízení neběží, je na klientech tato adresa nastavena na `127.0.0.1`.

Z výše uvedeného důvodu je hra omezena pouze na lokální sítě. Tedy jeden z hráčů v lokální síti musí na svém počítači spustit aplikaci realizující server a říci ostatním hráčům svoji IP adresu. Pro úspěšné připojení musí ostatní hráči danou adresu zadat do obrazovky s nastavením a stisknout tlačítko *OK*.

4.4 Synchronizace hry se serverem

Níže jsou popsána vlákna, která zajišťují komunikaci se serverem, probíhající v periodických časových intervalech. Využití těchto vláken je ilustrováno na obrázku 10. Počítačový klient k opakovanému volání metody, pomocí které komunikuje se serverem, využívá třídu `Timeline`[2], zatímco u Androidu je použita třída `AsyncTask`[3]. Každé z těchto vláken buď od serveru zjišťuje nebo mu zasílá určitou informaci.

Vlákno zjišťující jména ostatních hráčů
Pokud jsou na serveru uložena jména všech hráčů z dané hry, tak si je klient stáhne, aktualizuje grafické rozhraní a toto vlákno skončí.

Vlákno zjišťující, zda je daný hráč na tahu
Ptá se serveru, zda je na tahu právě daný hráč, pokud ano, umožní hráči položit kostičku na desku. Toto vlákno nabývá tří stavů:

- *Beží* – vlákno běží, pokud server odpovídá negativně, tj. daný hráč ještě není na tahu.
- *Pozastaveno* – v případě, že daný hráč je na tahu.
- *Zrušeno* – pokud daný hráč už nemá možnost, jak pokračovat ve hře.

Vlákno, které aktualizuje hrací desku
Stahuje od serveru aktuální informace o hrací desce, nabývá tří stavů:

- *Beží* – pokud daný hráč není na tahu.
- *Pozastaveno* – v případě, že daný hráč je na tahu.
- *Zrušeno* – pokud žádný hráč už nemá možnost, jak pokračovat ve hře.

Vlákno, které udržuje spojení
Úkolem tohoto vlákna je poskytovat serveru informaci o tom, zda je daný hráč stále připojen. V případě výpadku spojení na straně klienta dokáže server tento výpadek zaznamenat a daného hráče od hry odpojit. Vlákno ukončí svoji činnost v tom případě, když žádný hráč nemá možnost, jak pokračovat ve hře.

[2] https://docs.oracle.com/javase/8/javafx/api/javafx/animation/Timeline.html
[3] https://developer.android.com/reference/android/os/AsyncTask.html

4.5 Čekání na ostatní hráče

Čekání nastává, pokud se hráč připojí do hry po síti, ve které ještě není připojen požadovaný počet hráčů. Například v situaci, kdy daná hra je vytvořená pro 4 hráče a aktuálně do ní jsou připojeni 2 hráči. V tomto případě se spustí vlákno, které se periodicky dotazuje serveru, zda už se do hry připojili všichni hráči. Nabývá dvou stavů:

- *Beží* – pokud server odpovídá negativně, tzn. do hry se ještě nepřipojil požadovaný počet hráčů
- *Zrušeno* – pokud se do hry připojili všichni hráči, nebo jej zrušil uživatel.

Jakmile server jednou odpoví kladně, tedy jsou do hry připojeni všichni hráči, přejde se na obrazovku s hrou.

4.6 Odesílání a přijímání zpráv

Při hře po síti je nutné synchronizovat důležitá data se serverem. Data jsou synchronizována pomocí tzv. dotazů, viz 5.1. Odesílání dotazů na server, popřípadě přijímání odpovědí, mají u klienta na starost dvě níže popsané třídy.

Třída ClientQuery

Třída `ClientQuery` má za úkol odeslat po komunikačním kanálu dotaz na server. Dotaz je po komunikačním kanálu zaslán v podobě řetězce. Pokud je od serveru očekávána odpověď, vyčká na přijetí odpovědi a tu poskytne třídě `Client`, viz níže. Na odeslání dotazu a případné přijmutí odpovědi od serveru je stanoven časový interval dvou sekund. Pokud se do té doby nepodaří na server připojit, je třídě `Client` navrácena hodnota `null`.

Třída Client

Tato třída využívá třídu `ClientQuery` pro zasílání jednotlivých dotazů na server. Tyto dotazy jsou zde implementovány jako samostatné metody.

Kontrola konektivity

Pro kontrolu připojení ze strany klienta je využíváno počítadlo, které je při každém úspěšném připojení nastaveno na hodnotu 0. Naopak, při každém neúspěšném připojení je počítadlo inkrementováno.

Návratové hodnoty

Metody třídy `Client` můžeme rozdělit na dva typy, podle toho, zda daná metoda očekává odpověď od serveru:

- **Očekávává odpověď** – pokud je od serveru očekávána odpověď, tak daná metoda vrací dvojici hodnot:
  - Odpověď od severu.

  - Hodnotu počítadla neúspěšných připojení.

- **Neočekává odpověď** – vrací pouze hodnotu počítadla neúspěšných připojení.

Klient toleruje neúspěšná připojení, dokud jejich počet nepřekročí pevně danou hranici. Jinak je hráči signalizováno jeho odpojení od hry.

4.7 Uložení hry

Při ukládání hry u počítačového klienta se vytvoří nová složka v domovském adresáři uživatele, kam se následně všechny hry ukládají. Na systému Windows vypadá cesta k uloženým hrám následovně:

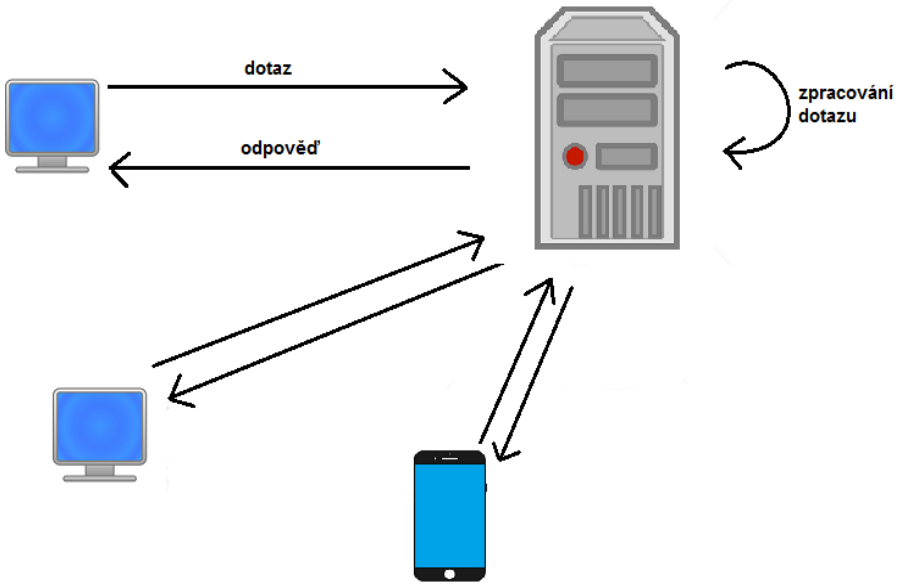
```
C:\Users\Uzivatel\Blokus\Save\jmeno.xml
```

Na systémech Linux a macOS je struktura adresářů pro uložení hry stejná.
Klient pro platformu Android dané soubory ukládá do adresáře, který reprezentuje interní úložiště pro aplikaci Blokus.

5 Server

Při hraní hry po síti spolu musí všichni klienti navzájem sdílet data, aby se zajistil plynulý průběh hry. Protože přeposílání všech dat mezi jednotlivými klienty by bylo složité a zbytečné, tak veškerá komunikace probíhá přes server, viz obrázek 12. Ten představuje jakéhosi prostředníka, na který klienti posílají svoje dotazy, respektive požadavky. Jedná se tedy o tzv. dotazovací server.

Pokud vynecháme implementační detaily, tak server zprostředkovává klientovi prostor, kam zapisovat všechny informace o dané hře. Těmito informacemi je myšleno například provedení tahu nebo odpojení se od hry. Všichni klienti aktualizují svoji obrazovku s hrou na základě informací získaných od serveru a následné změny opět posílají na server. Tímto způsobem je zajištěn přenos informací mezi všemi klienty.



Obrázek 12: Ilustrace komunikace klientů se serverem

5.1 Komunikace s klienty

Pokud klient zasílá na server nějakou informaci nebo o ni naopak žádá, posílá serveru tzv. dotaz. Zaslání/získání informace probíhá ve třech fázích. Nejprve klient zašle na server dotaz, ten ho zpracuje a následně odešle případnou odpověď. Pro komunikaci mezi serverem a klienty je využíván síťový protokol TCP/IP.

5.1.1 Struktura dotazu

Dotaz je typu řetězec a skládá se ze dvou částí:

- **Identifikátor dotazu** – sekvence zpravidla 2-3 písmen, tento identifikátor určuje, jakou informaci klient od serveru požaduje nebo naopak, jakou zasílá. Na základě tohoto identifikátoru server zavolá konkrétní metodu.
- **Parametry dotazu** – jednotlivé hodnoty, které server předá jako parametry volané metodě.

Identifikátor a jednotlivé parametry jsou odděleny mezerou. Dotaz tedy v obecném tvaru vypadá následovně:

$$"identifikátor␣parametr_1␣\ldots␣parametr_n"$$

Všechny identifikátory, jim odpovídající metody a potřebné argumenty jsou uvedeny v tabulce 1.
Třída `ServerMethods`, jejíž metody jsou volány na základě jednotlivých identifikátorů, je popsána v kapitole 6.4.

Protože jednotlivé části dotazu jsou oddělené mezerou, je uživatelům zakázáno používat znak mezery v synchronizovaných datech, tj. v názvu hry nebo jménu hráče.

Tabulka identifikátorů		
Identifikátor	**Parametry**	**Volaná metoda třídy `ServerMethods`**
CGR	String name, int requiredPlayers, int countOfPcOpp, String blockBricks	`createGame`
FPR	int gameID	`findPlayerst`
ACH	int gameID, int playerID	`aliveCheck`
GFS	int gameID	`getFinalScore`
DIS	int gameID, int playerID	`disconnect`
RGR	String name	`removeGame`
GUP	int gameID	`getUpdate`
GAG	$\emptyset$	`getAvailibleGames`
GT	int gameID, playerID	`getToken`
JG	String name	`joinGame`
APN	String name, int gameID, int playerID	`addPlayerName`
GPN	int gameID	`getPlayersName`
INR	int gameID	`isNamesReady`
GMR	int gameID, int playerID, int[] fieldsID	`gameMove`
AO	String name	`isAllOut`
IMO	int gameID, int playerID	`imOut`
IGA	String name	`isGameAvailible`

Tabulka 1: Identifikátory dotazů

5.1.2 Zpracování dotazu

Zpracování dotazu na straně serveru je rozděleno na tři logické části.

- **Navázání spojení** – mezi klientem a serverem je navázáno spojení, server přečte z komunikačního kanálu řetězec zaslaný klientem.
- **Zjištění požadavku** – přijatý řetězec je rozdělen na jednotlivé části, z těchto částí je získán identifikátor dotazu a jeho parametry.
- **Obsluha požadavku** – je zavolána metoda odpovídající danému identifikátoru s danými parametry. Následně je odeslána případná odpověď získána zavoláním dané metody.

6 Implementace serveru

6.1 Zpracování dotazu

Logické rozdělení na tři části, viz výše, je implementováno pomocí čtyř tříd.

- `Server`[4] – naváže spojení, přečte zprávu, rozpozná identifikátor dotazu a zavolá konkrétní metodu třídy `ServerMethods`. Zavolané metodě je předána zpráva od klienta jako parametr.
- `ServerMethods` – z předaného řetězce zjistí parametry dotazu a zavolá konkrétní metodu třídy `ServerGameOrganizer` s danými parametry.
- `ServerGameBoard` a `ServerGameOrganizer` – zjišťují požadovanou informaci nad konkrétní hrou.

6.2 Třída ServerGameBoard

Třída reprezentující právě jednu síťovou hru. Uchovává si všechny podstatné informace o dané hře. Hlavní funkčností této třídy je poskytování a zpracovávání informací od jednotlivých klientů. Mezi nejdůležitější informace uchovávané třídou ServerGameBoard patří:

- Počet hráčů.
- Hrací deska.
- Počet již přihlášených hráčů.
- Počet počítačových hráčů.
- Identifikátor (ID) dané hry.
- Seznam hráčů, kteří jsou již mimo hru.
- Kdo je další na tahu.
- Seznam hráčů, kteří se již odpojili od hry.
- Tabulka, ve které se udržují informace o tom, který hráč má jaké jméno.
- Zda se daná hra již hraje, nebo se čeká na připojení zbývajících hráčů.

[4] Zde je myšleno jako třída

6.2.1 Důležité funkce

Poskytnutí jmen všech připojených hráčů
Po startu hry po síti všichni klienti odešlou informace o jménech daných hráčů na server. Server postupně tyto jména společně s ID příslušným daným hráčům zapisuje do výše zmíněné tabulky. Pokud tabulka obsahuje informace o jménech všech hráčů, tak si je jednotliví klienti stáhnou a aktualizují grafické rozhraní.

Určení hráče, který je na tahu
Při vytvoření hry po síti dostanou všichni hráči jednoznačný identifikátor (ID).

K určení hráče, který je na tahu, slouží počítadlo, které je při vytvoření hry po síti inicializováno na hodnotu 1. To znamená, že další na tahu je hráč s ID = 1. Pokud tento hráč provede svůj tah, zašle o tom serveru informaci a ten inkrementuje počítadlo na 2.

Pro příklad předpokládejme, že daná hra je pro 4 hráče. Pokud by hráč s ID = 3 byl již mimo hru, tak server po tahu hráče s ID = 2 inkrementuje počítadlo rovnou na hodnotu 4. Po tahu čtvrtého hráče je počítadlo nastaveno na 1. Tento cyklus se opakuje, dokud alespoň jeden hráč může pokračovat ve hře.

Poskytování aktuální hrací desky
Při vytvoření hry po síti je inicializována hrací deska pro požadovaný počet hráčů. Po obdržení informace o tahu daného hráče se zapíše tento tah na hrací desku. Díky tomu, že hráč dostane povolení k tahu až poté, co předchozí hráč zašle informace o svém tahu na server, je danému hráči vždy poskytnuta aktuální deska.

Provádění tahů počítačových hráčů
Při vytvoření hry po síti s aktivními počítačovými hráči dostanou tito hráči jednotlivá ID. Pomocí tohoto ID s nimi počítadlo určující dalšího hráče na tahu, pracuje stejně jako s lidskými hráči. Pokud je na tahu počítačový hráč, tak je serverem vyzván k provedení tahu. Poté se vyčká na obdržení informace o tahu počítačového hráče a hra pokračuje. Způsob výpočtu tahu počítačového hráče je popsán v kapitole 7.

Určení konce hry
Server si průběžně udržuje seznam hráčů, kteří jsou již mimo hru. Do tohoto seznamu jsou postupně přidáváni hráči na základě dvou faktorů:

- Server dostane od klienta informaci o tom, že daný hráč už nemůže pokračovat.
- Pokud se klient serveru dlouho nepřihlásí, viz níže, je hráč přidán do tohoto seznamu.

Kontrola připojených hráčů
Třída `ServerGameBoard` obsahuje vlákno kontrolující, zda se některý z připojených hráčů nedopatřením neodpojil (například výpadkem spojení). Pokud tedy

u klienta dojde k výpadku spojení, server tento výpadek zaregistruje, daného hráče od hry odpojí a přidá ho do seznamu hráčů, kteří již nemohou pokračovat ve hře.

6.3 Třída ServerGameOrganizer

Třída `ServerGameOrganizer` slouží k organizaci více her po síti najednou. Při dotazu od klienta přistoupí ke konkrétní hře a provede nad ní požadovanou operaci. Tato třída obsahuje:

- **Seznam síťových her** – každá hra v tomto seznamu je reprezentována třídou `ServerGameBoard` a má jednoznačné ID, pomocí kterého k ní lze přistupovat.
- **Tabulku, která mapuje ID dané hry na její jméno** – upravený seznam jmen z této tabulky je poskytován klientům, pokud požádají o hry, do kterých se lze připojit.

Výčet metod třídy ServerGameOrganizer

- `public void aliveCheck(int playerID, int gameID)`
  Podá hře určené parametrem `gameID` informaci o tom, že hráč určený parametrem `playerID` je stále aktivně připojen.
- `public void removeGame(String name)`
  Hru určenou parametrem `name` odstraní jak ze seznamu běžících her, tak z tabulky mapující ID hry na její jméno.
- `public int newGame(String name, int requiredPlayers, int countOfPcOpp, String blockBricks)`
  Přidá do seznamu her novou síťovou hru s parametry:
  - `name`: jméno hry,
  - `requiredPlayers`: celkový počet hráčů,
  - `countOfPcOpp`: počet počítačových hráčů,
  - `blockBricks`: zda se jedná o mód s překážkami.
- `public void clearUnactiveGames()`
  Odstraní všechny dohrané hry ze seznamu, to jsou takové, do kterých není připojen žádný hráč.
- `public boolean isAllOut(String name)`
  Vrací `true`, pokud jsou všichni hráči ve hře určené parametrem `name` mimo hru, jinak vrací `false`.
- `public void outPlayer(int gameID, int playerID)`
  Hře určené parametrem `gameID` podá informaci o tom, že hráč určený parametrem `playerID` už nemá možnost pokračovat.

- `public void insert(int gameID, int playerID, int[] fieldID)`
  Na hrací desku hry určené parametrem `gameID` zapíše na políčka určené parametrem `fieldID` hodnotu `playerID`.

- `public String getUpdate(int gameID)`
  Vrací hrací desku hry, určené parametrem `gameID`, zakódovanou do řetězce.

- `public void addName(int gameID, int playerID, String name)`
  Hře určené parametrem `gameID` podá informaci o tom, že hráč určený parametrem `playerID`, má v dané hře jméno dané parametrem `name`.

- `public boolean isNamesReady(int gameID)`
  Vrací `true`, pokud všichni hráči připojení do hry určené parametrem `gameID` zaslali svá jména.

- `public String getNamesByPlayersID(int gameID)`
  Vrací jména všech hráčů, kteří jsou připojeni do hry určené parametrem `gameID`, zakódované do řetězce.

- `public String getAvailibleGames()`
  Vrací seznam všech síťových her, do kterých se lze připojit, zakódovaný do řetězce.

- `public void disconnect(int gameID, int playerID)`
  Hře určené parametrem `gameID` podá informaci o tom, že hráč určený parametrem `playerID` se odpojil.

- `public String getFinalScore(int gameID)`
  Zkontroluje, zda ve hře určené parametrem `gameID`, má nějaký z hráčů nárok na přičtení bonusů. Poté v řetězci vrátí jednotlivé počty bodů získané danými hráči.

- `public boolean isGameAvailible(String name)`
  Vrací `true`, pokud hra určená parametrem `name` existuje a lze se do ní připojit, jinak vrací `false`.

6.4 Třída ServerMethods

Funguje nad třídou ServerGameOrganizer, využívá její metody a na základě získaných informací poskytuje klientům odpovědi. Všechny metody této třídy mají dva parametry:

- **Zprávu od klienta** – z tohoto řetězce jsou získány potřebné parametry pro zavolání dané metody třídy `ServerGameOrganizer`.

- **Stream** – po kterém se zašle případná odpověď.

Popis a výčet metod třídy ServerMethods
V této sekci označme třídu `ServerGameOrganizer` zkratkou `sgo`.

- `public void isAllOut(String line, DataOutputStream dos)`
  Zavolá metodu `isAllOut` třídy `sgo` a na základě vrácené hodnoty zašle zprávu klientovi.

- `public void imOut(String line, DataOutputStream dos)`
  Zavolá metodu `outPlayer` třídy sgo.

- `public void isNamesReady(String line, DataOutputStream dos)`
  Na základě vrácené hodnoty metodou `isNamesReady` třídy `sgo` zašle odpověď klientovi.

- `public void getPlayersName(String line, DataOutputStream dos)`
  Jako odpověď zašle klientovi řetězec získaný zavoláním metody `getNamesByPlayerID` z třídy `sgo`.

- `public void addPlayerName(String line, DataOutputStream dos)`
  Zavolá metodu `addName` třídy `sgo`.

- `public void getToken(String line, DataOutputStream dos)`
  Nevolá žádnou metodu třídy `sgo`. Nejprve získá z řetězce `line` parametry `gameID` a `playerID`. Následně odešle klientovi informaci o tom, zda hráč určený parametrem `playerID` je ve hře určené parametrem `gameID` na tahu.

- `public void gameMove(String line, DataOutputStream dos)`
  Zavolá metodu `insert` třídy `sgo`.

- `public void createGame(String line, DataOutputStream dos)`
  Zavolá metodu `newGame` třídy `sgo`.

- `public void findPlayers(String line, DataOutputStream dos)`
  Nevolá žádnou metodu třídy `sgo`. Nejprve získá z řetězce `line` parametr `gameID` a klientovi odešle informaci o tom, zda se do hry určené tímto parametrem připojili již všichni hráči.

- `public void joinGame(String line, DataOutputStream dos)`
  Nevolá žádnou metodu třídy `sgo`. Nejprve získá z řetězce `line` parametr `name`, následně odešle klientovi informaci o tom, zda byl do hry určené parametrem `name` připojen.

- `public void disconnect(String line, DataOutputStream dos)`
  Zavolá metodu `disconnect` třídy `sgo`.

- `public void aliveCheck(String line, DataOutputStream dos)`
  Zavolá metodu `aliveCheck` třídy `sgo`.

- `public void getAvailibleGames()`
  Zavolá metodu `getAvailibleGames` třídy `sgo`.

- `public void removeGame(String line, DataOutputStream dos)`
  Zavolá metodu `removeGame` třídy `sgo`.

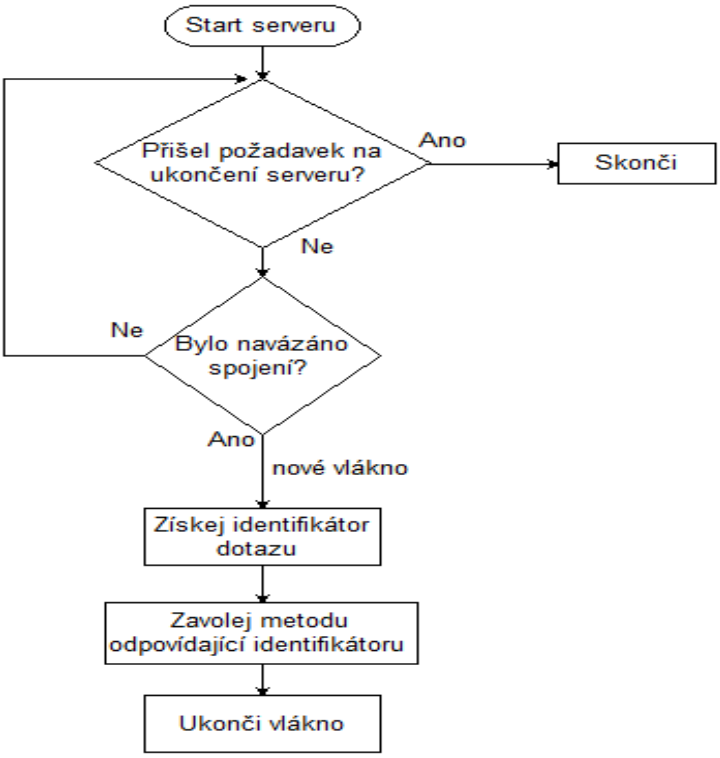
- `public void getUpdate(String line, DataOutputStream dos)`
  Klientovi odešle aktuální desku získanou zavoláním metody `getUpdate` třídy `sgo`.

- `public void getFinalScore(String line, DataOutputStream dos)`
  Klientovi odešle konečné skóre dané hry, získané zavoláním metody `getFinalScore` třídy `sgo`.

- `public void isGameAvailible(String line, DataOutputStream dos)`
  Zavolá metodu `isGameAvailible` třídy `sgo`.

6.5 Třída Server

Třída `Server` obsahuje tabulku, kde jsou na jednotlivé identifikátory dotazů namapovány konkrétní metody třídy `ServerMethods`, viz 6.4. Tato třída čeká na příchozí spojení, jakmile je spojení navázáno, je pro dané spojení vytvořeno nové vlákno. V tomto vláknu je nejprve přečten dotaz od klienta. Z tohoto dotazu je získán jeho identifikátor a následně je zavolána na něj namapovaná metoda. Diagram zpracování dotazu ze strany serveru je uveden na obrázku 13.



Obrázek 13: Zpracování dotazu na straně serveru

6.6 Komunikace

K realizaci architektury typu klient-server je využita třída `Socket`[5]. Pro zasílání dotazů/odpovědí jsou využity streamy `DataInputStream`[6] a `DataOutputStream`[7]. Pokud zasílá server klientovi odpověď, tak je daná odpověď převedena na pole bytů. Odeslání pole bytů proběhne tak, že server nejprve odešle po streamu velikost tohoto pole a následně pole samotné. Velikost je klientovi zasílána proto, aby věděl, kolik bytů má ze vstupního streamu přečíst, a předešlo se chybám.

[5] https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html
[6] https://docs.oracle.com/javase/7/docs/api/java/io/DataInputStream.html
[7] https://docs.oracle.com/javase/7/docs/api/java/io/DataOutputStream.html

7 Počítačový hráč

Počítačového hráče v této aplikaci představuje třída `PcPlayer`. Při inicializaci je počítačovému hráči předána sada kostiček jedné barvy. Pro provedení tahu používá počítačový hráč algoritmus uvedený v sekci 7.1.

Počítačového hráče lze kromě hry proti počítači využít i ve hře po síti. Rozdíl je v tom, že pokud uživatel zvolí hru pouze proti počítači, probíhá výpočet dalšího tahu počítačového hráče na daném klientu, zatímco u hry po síti zajišťuje výpočet server, konkrétně třída `ServerGameBoard`.

7.1 Algoritmus

Při tvorbě počítačového hráče bylo využito principu algoritmu Minimax ze zdroje informací [3]. Algoritmus pro výpočet tahu počítačového hráče nejdříve vygeneruje všechny možnosti, jak by mohl další tah provést. Vygenerované možnosti jsou ořezány na základě níže uvedených kritérií. Z možností, které zbyly, je vybrána ta nejlepší. Algoritmus je popsán pseudokódem 1. Všechny použité metody a kritéria, podle kterých je nejlepší možnost vybírána, jsou popsány níže.

Proměnné využité v pseudokódu:

- `int turnCount` – značí počet již provedených tahů počítačovým hráčem, inicializován na `0`
- `boolean imOut` – pravdivostní hodnota nesoucí informaci o tom, zda je daný počítačový hráč již mimo hru, inicializována na `false`
- `List<PcMoveOption`[8]`>options` – seznam, do kterého se ukládají generované možnosti
- `PcMoveOption best` – zvolená možnost, kterou počítačový hráč použije pro svůj tah

[8] Třída pro reprezentaci tahu počítačového hráče

```
if turnCount < 3 then
    makeFirstMoves();
    turnCount++;
    return;
else
    options := generate(board, bricks, playerID,
     maxSize);
    if options = ∅ then
        imOut := true;
        return;
    end
    options := trimOptions(options, enemies, bricks,
     trimSize);
    best := findBestOption(board, options, enemies,
     bricks, level, playerID);
    turnCount++;
end
```

Algorithm 1: Výpočet tahu počítačového hráče

Popis algoritmu

Jako první se ověří podmínka, zda daný počítačový hráč provedl více jak tři tahy. Pokud ne, využije se metoda `makeFirstMoves`, viz 7.2.5. Ta zajišťuje první tři tahy počítačového hráče. Po provedení jednoho z prvních tří tahů je inkrementováno počítadlo odpovídající počtu odehraných kol a algoritmus skončí.

Pokud už jsou odehrána více jak tři kola, postupuje algoritmus následovně:

1. Pomocí metody `generate`, viz 7.2.3, se vygenerují všechny možnosti, jak by mohl počítačový hráč provést svůj příští tah.

2. Ověří se, zda seznam všech vygenerovaných možností není prázdný. Pokud prázdný je, znamená to, že daný počítačový hráč už nemá možnost, jak pokračovat. Proměnná `imOut` je tedy nastavena na hodnotu `true` a algoritmus skončí.

3. Ze seznamu všech vygenerovaných možností jsou vybrány takové, které mají největší hodnotu kostičky nebo blokují nejvíce nepřítel, viz 7.2.4.

4. Z možností vybraných v bodě 3 je zvolena ta nejlepší pomocí metody `findBestOption`, viz 7.2.6.

5. Inkrementuje se počítadlo odpovídající počtu odehraných kol a algoritmus skončí.

7.2 Metody využívané počítačovým hráčem

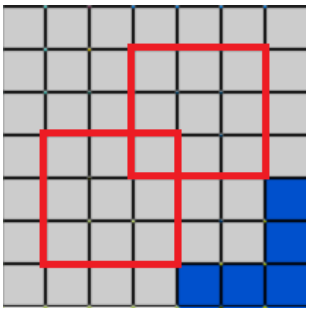
V této kapitole jsou popsány metody uvedené v pseudokódu 1 a ostatní důležité metody, které slouží pro výpočet tahu počítačového hráče.

7.2.1 Metoda recognizeMiniSquares

Parametry:

- `int playerID` – identifikátor hráče
- `Set<Point> cornersCoordinates` – struktura pro výsledky
- `GameBoard board` – aktuální hrací deska
- `int maxSize` – rozměr největší dostupné kostičky daného hráče

Popis
Metoda `recognizeMiniSquares` není přímo použita v pseudokódu, ale využívá jí metoda `findAllOptions`. Slouží k nalezení souřadnic všech políček na hrací desce, odkud by mohl počítačový hráč provést tah. Počet těchto políček přímo závisí na parametru `maxSize`. Na obrázku 14 jsou červeně ohraničena políčka, jejichž souřadnice tato metoda vrátí pro `maxSize = 3` a ilustrovaný stav hrací desky.



Obrázek 14: Vizualizace metody recognizeMiniSquares

7.2.2 Metoda findAllOptions

Parametry:

- `Set<Point> cornersCoordinates` – Množina souřadnic políček na hrací desce vrácená metodou `recognizeMiniSquares`
- `List<Brick> bricks` – dostupné kostičky daného hráče
- `GameBoard board` – aktuální hrací deska
- `int playerID` – identifikátor hráče

Popis
Metoda `findAllOptions` je využívána metodou `generate`. Vrací seznam všech vygenerovaných možností, jak lze ještě dostupné kostičky umístit na desku. Tedy pro každé políčko v množině `cornerCoordinates` vyzkouší všechny

možné rotace všech dostupných kostiček. Jakmile lze zkoušenou variantu umístit na desku, je tato varianta přidána do seznamu vygenerovaných možností.

Optimalizace na platformě Android
Metoda `findAllOptions` je časově nejsložitější ze všech metod, které počítačový hráč používá. Protože mobilní telefony jsou výkonnostně slabší než osobní počítače, je tato metoda pro platformu Android upravena tak, že prochází kostičky od takových, které mají největší cenu a po vygenerování prvních 200 možností skončí.

7.2.3 Metoda generate

Parametry:

- `GameBoard board` – aktuální hrací deska
- `List<Brick> bricks` – dostupné kostičky daného hráče
- `int playerID` – identifikátor hráče
- `int maxSize` – rozměr největší dostupné kostičky

Popis
Vrací všechny možnosti dalšího tahu počítačového hráče vzhledem k dostupným kostičkám a aktuálnímu stavu hrací desky. Postupuje následovně:

1. Nejdříve získá souřadnice takových bodů na hrací desce, odkud by mohl počítačový hráč provést svůj další tah, tj. zavolá metodu `recognizeMiniSquares`.
2. Se získanými souřadnicemi zavolá metodu `findAllOptions` a vrátí její výsledek.

7.2.4 Metoda trimOptions

Parametry:

- `List<PcMoveOption> options` – seznam všech vygenerovaných možností vrácený metodou `generate`
- `List<Integer> enemies` – souřadnice rohů všech ostatních hráčů
- `List<Brick> bricks` – dostupné kostičky daného hráče
- `int trimSize` – velikost, na kterou se má seznam možností ořezat

Ořezávací kritéria:

- Hodnota kostičky.

- Počet zablokovaných rohů nepřátel.

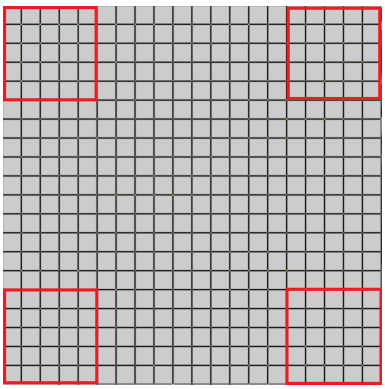
Popis
Z možností předaných parametrem `options` je vybráno nejlepších `trimsize` na základě každého ořezávacího kritéria. Tedy ve výsledku může metoda vracet až `2 * trimSize` možností.

7.2.5 Metoda makeFirstMoves

Parametry: ∅
Popis
Slouží k provedení prvních tří tahů počítačového hráče. O prvním tahu můžeme říci že:

- Každý hráč musí dle pravidel začít v jenom z rohů.
- Rozměr největší dostupné kostičky pro každého hráče je roven pěti, viz obrázek číslo 1.

Na základě těchto údajů je pro provedení prvního tahu zavolána metoda `findAllOption`, kde se jako parametr `cornersCoordinates` předá jedna z množin bodů ohraničených na obrázku 15. Pro druhý a třetí tah je již zavolána metoda `generate`.



Obrázek 15: Vizualizace výběru prostoru pro první tah počítačového hráče

Výběr nejlepší možnosti
Nejlepší možnost z prvních tří tahů je vybrána na základě:

- Směru po diagonále ke středu hrací desky.
- Hodnoty kostičky.

7.2.6 Metoda findBestOption

Parametry:

- `GameBoard board` – aktuální hrací deska
- `List<PcMoveOption> options` – seznam ořezaných možností vrácený metodou `trimOptions`
- `List<Integer> enemies` – souřadnice rohů všech ostatních hráčů
- `List<Brick> bricks` – dostupné kostičky daného hráče
- `int level` – určuje hloubku generovaného stromu, viz níže
- `int playerID` – identifikátor hráče

Popis
Pro každou možnost ze seznamu `options` metoda simuluje budoucích `level` tahů, které by mohl počítačový hráč provést. Tedy pro každou možnost ze seznamu `options` je rekurzivně vygenerován strom tahů, které by v budoucnu připadaly v úvahu, pokud by hráč zvolil danou možnost. Strom má hloubku `level`. Každý uzel vygenerovaného stromu představuje jeden budoucí tah pro danou možnost a obsahuje informace o tom, kolik nepřátelských rohů dokáže tímto tahem zablokovat.

Výběr nejlepší možnosti
Po průchodu vygenerovaným stromem jsou pro každou možnost ze seznamu `options` k dispozici následující fakta:

- Hodnota kostičky užité v dané možnosti.
- Kolik nepřátelských rohů dokáže hráč zablokovat v příštích `level` tazích, pokud zvolí danou možnost.

Po průchodu vygenerovaným stromem může být počet nepřátelských rohů, které lze v budoucích tazích zablokovat:

- **roven 0**
  Pak je jako nejlepší vybrána jedna z možností, které mají největší hodnotu pokládané kostičky.
- **větší než 0**
  Jako nejlepší je zvolena možnost s největším počtem zablokovaných rohů v budoucích tazích.

8 Použité technologie

Java FX
Java FX je softwarová platforma na bázi platformy Java. Umožňuje vytvářet multiplatformní aplikace. Podporuje kombinaci různých grafických prvků, ať už jde o audio, video či animace. Velkou výhodou platformy Java FX je spustitelnost aplikací na mnoha typech zařízení, jako jsou počítače, tablety, chytré telefony nebo televize.

Cascading Style Sheets - CSS
V překladu kaskádové styly, jedná se o jazyk popisující styl jednotlivých prvků grafického rozhraní aplikace. Výhodou CSS je oddělení vzhledu aplikace od její programátorské podstaty.

Eclipse IDE
Eclipse je v současnosti nejrozšířenější vývojové prostředí pro platformu Java. Jedná se o open source s vysokou rozšiřitelností. Do vývojového prostředí lze nainstalovat mnoho pluginů, díky kterým lze rozšířit seznam podporovaných jazyků například o C++ nebo PHP. Po stažení určitých pluginů lze v Eclipse vyvíjet i aplikace pro platformu Android.

Android Studio IDE
Vývojové prostředí pro aplikace na platformě Android. Je relativně nové, bylo představeno v roce 2013. Obsahuje integrovaný emulátor mobilního zařízení. Android Studio postupně nahrazuje Eclipse ve vývoji aplikací pro Android, hlavně díky mnohem jednoduší instalaci a rychlosti.

Závěr

V rámci bakalářské práce byla v jazyku Java implementována hra Blokus s režimem hry po síti a nativním klientem pro více platforem. Práce se skládá ze tří částí, klientu pro počítač, klientu pro Android a serveru. Aplikace by mohla být do budoucna rozšířena například o další varianty hry nebo volbu obtížnosti počítačového hráče.

Conclusions

In the bachelor thesis was implemented game Blokus in the Java language with network mode and a native client for multiple platforms. The work consists of three parts, a client for a computer, an Android client and a server. The application could be expanded in the future by, for example, other variants of the game or by choosing the difficulty of a computer player.

A Zprovoznění klientů

A.1 Počítačový klient

Systémové požadavky
Počítačový klient nemá na systém vysoké požadavky, nicméně pro plynulý běh celého klienta se doporučuje:

- Paměť RAM - 1GB a více
- Minimální rozlišení monitoru - 1024x768

Počítačový klient byl testován na 64-bit. systémech:

- Windows 7 a vyšší
- Debian verze 7.11
- macOS 10.12 Sierra

Spuštění počítačového klienta
Protože počítačový klient je napsán v jazyce Java, konkrétně Java 8, je pro jeho spuštění nutná instalace příslušného Oracle SDK 8. Všechny SDK pro příslušné platformy jsou dostupné z adresy: https://goo.gl/6h29TW. Po instalaci SDK je už možné klienta spustit. Počítačový klient se nachází na přiloženém CD v adresáři `bin_pc/` pod jménem `blokus.jar`.

A.2 Klient pro Android

Systémové požadavky
Mobilní aplikace byla vyvíjena pro API 19, tedy Android 4.4 (KitKat) a vyšší. Co se týče technických parametrů, tak se pro plynulý běh klientu na mobilním zařízení doporučuje:

- Paměť RAM - 1GB a více

Mobilní aplikace byla testována na zařízeních:

- Lenovo Vibe X2
- LG G4
- Lenovo P70
- Lenovo Tab 2 A8-50F

Instalace klienta pro Android
Aby bylo možné klienta na mobilní zařízení nainstalovat, je nejdříve třeba na daném mobilním zařízení povolit instalaci z neznámých zdrojů. To je dostupné přes: `menu > nastavení > zabezpečení > neznámé zdroje`. Samotná instalace probíhá v následujících krocích:

- Připojení mobilního zařízení k počítači.
- Zkopírovaní instalačního souboru `blokus.apk` nacházejícího se na přiloženém CD v adresáři `bin_android/` do některé složky na mobilním zařízení, například `Downloads`.
- Odpojení mobilního zařízení od počítače.
- Nalezení instalačního souboru na mobilním zařízení a spuštění instalace.

Po provedení těchto několika kroků je možné spustit hru Blokus z menu příslušného mobilního zařízení.

B Zprovoznění serveru

Systémové požadavky
Pro plynulý běh serveru při více síťových hrách se doporučuje:

- Paměť RAM - 2GB a více.

Server byl testován pouze v lokálních sítích na zařízení s:

- Operačním systémem Windows 7, 64-bit
- Pamětí RAM - 4GB
- Procesorem - Intel Core i5 (2.53GHz)

Spuštění serveru
Opět je nutné stáhnout potřebné Oracle SDK 8, viz výše. Server se spouští v příkazové řádce. Nejdříve je třeba zkopírovat z přiloženého CD soubor `blokus_server.jar` do počítače.

U systému Windows tedy například do adresáře:

`C:\Users\Uzivatel\Blokus\Server`.

Samotné spuštění se pak provede následujícími kroky:

- Spuštění příkazové řádky
- Přepnutí do adresáře, kam je nakopírován soubor `blokus_server.jar`, vzhledem k předchozímu adresáři příkazem:

```
cd C:\Users\Uzivatel\Blokus\Server
```

- Spuštění samotné aplikace realizující server, příkazem:

```
java -jar blokus_server.jar
```

Ukončení činnosti serveru klávesovou zkratkou `Ctrl + c`

C Obsah přiloženého CD

bin_pc/
: Spustitelný počítačový klient hry Blokus a server nutný pro hru po síti.

bin_android/
: Instalační soubor klienta hry Blokus pro platformu Android.

doc/
: Text práce ve formátu PDF včetně zdrojových souborů potřebných pro vygenerování PDF dokumentu.

src/
: Kompletní zdrojové texty jednotlivých klientů hry Blokus a serveru.

Seznam literatury

[1] DEA, Carl, Mark HECKLER, Gerrit GRUNWALD et al. JavaFX 8 2nd Edition Book | Java Books and Manuals. Books on Programming & IT Free Download | ITBooksHub.com [online]. Copyright © 2015 [cit. 29.04.2017].
Dostupné z: http://itbookshub.com/javafx-8-2nd-edition-book

[2] HAROLD, Elliotte Rusty. Java Network Programming, 4th Edition - Free Download eBook - pdf. IT eBooks - Free Download - Big Library [online]. Copyright © 2011 [cit. 2.04.2017].
Dostupné z: http://www.it-ebooks.info/book/3137

[3] KÜHR, Tomáš. Algoritmy realizující počítačového hráče v jednoduchých deskových hrách. Informatika - Katedra informatiky na UPOL [online]. Copyright © 2017 Katedra informatiky [cit. 16.04.2017].
Dostupné z: http://www.inf.upol.cz/downloads/studium/PS/algoritmy.pdf

[4] MEDNIEKS, Zigurd R., Laird DORNIN, G. Blake MEIKE et al. Programming Android, 2nd Edition - Free Download eBook - pdf. IT eBooks - Free Download - Big Library [online]. Copyright © 2011 [cit. 29.04.2017].
Dostupné z: http://www.it-ebooks.info/book/1988

[5] ZECHNER, Mario. and Robert GREEN. Beginning Android Games, 2nd Edition | Free PDF Books. Free PDF Books - Download free PDF eBooks, Lectures Notes and eBooks [online]. Copyright © 2016 [cit. 10.04.2017].
Dostupné z: http://freepdf-books.com/beginning-android-games-2nd-edition

[6] Blokus – Wikipedie. [online].
Dostupné z: https://cs.wikipedia.org/wiki/Blokus