Západočeská univerzita v Plzni

Mobilní komunikace a zařízení

KIV/MBKZ

Dokumentace semestrální práce

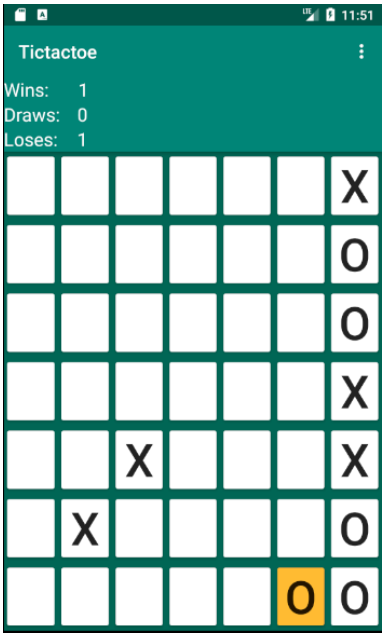
Vojtěch Daniš́ík
A19N0028P
danisik@students.zcu.cz

10. března 2020

Obsah

1 Zadání

Jako zadání semestrální práce pro předmět MBKZ byla vybrána hra **Piškvorky - TicTacToe**. Hlavní podstatou piškvorek je sestavit řadu o velikosti 5 po sobě jdoucích značek. Značky můžou být v diagonálních směrech, zleva doprava a odshora dolů. Hra se hraje ve dvou, kdy první hráč má značku **X** a druhý hráč má značku **O**. Za hráče se může považovat člověk i počítač. Hráči se snaží dosáhnout vítězné řady ale zároveň zabránit seskládání vítězné řady u protihráče. Vzhled této hry lze vidět na obrázku 1.



Obrázek 1: Piškvorky - ukázka

2 Programátorská dokumentace

V programu bylo použito pro docílení správné funkcionality 7 tříd.

Třída **MainActivity** slouží jako hlavní třída aplikace. V této třídě se děje zobrazování již vytvořených struktur jako jsou například vytváření tlačítek, hracího pole, přiřazování aktivit tlačítkům.

Třída **PlayerEngine** slouží pro zjištění tahu pro počítač. Pro zjištění tohoto tahu byl implementován algoritmus **Depth-First Search** algoritmus, který je ale omezen na maximální hloubku 4.

Třída **Constants** obsahuje veškeré konstanty použité v aplikaci.

Třída **Node** slouží jako jednoduchá reprezentace tlačítek při zjišťování tahu pro počítač.

Třída **CustomButton** je rozšířením základního tlačítka v android studiu. Obsahuje dodatečné informace při práci s hledáním v poli - souřadnice X a Y a také který hráč označil dané tlačítko.

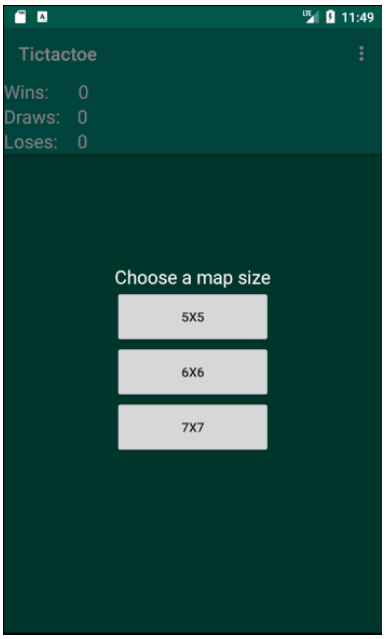
Třída **MainActivity** slouží jako hlavní třída aplikace. V této třídě se děje zobrazování již vytvořených struktur, jako jsou například časovač, hrací panel ale také i přiřazování aktivit pro tlačítka RESET a SET FLAG.

Třída **EGameStatus** je enum, která reprezentuje aktuální stav hry.

Třída **MoveChecker** slouží k zjištění, zda aktuální tah vedl k výhře hráče.

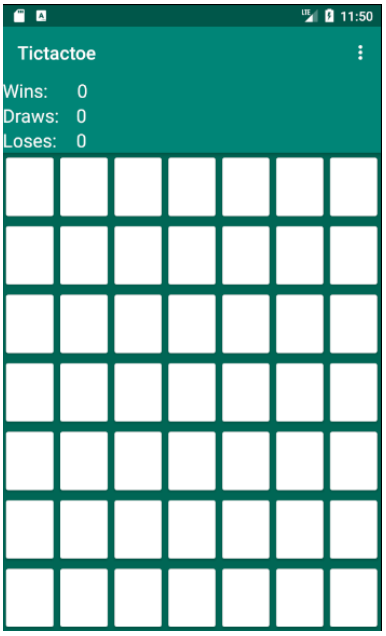
3 Uživatelská dokumentace

Výsledný vzhled aplikace lze vidět na několika obrázcích. První obrázek (viz 2) vyobrazuje aplikaci po jejím spuštění, kdy si hráč vybírá velikost mapy z tří zvolených možností (5x5, 6x6, 7x7). Aplikace navíc obsahuje počítadla výher/remíz/proher prvního hráče (Vás).



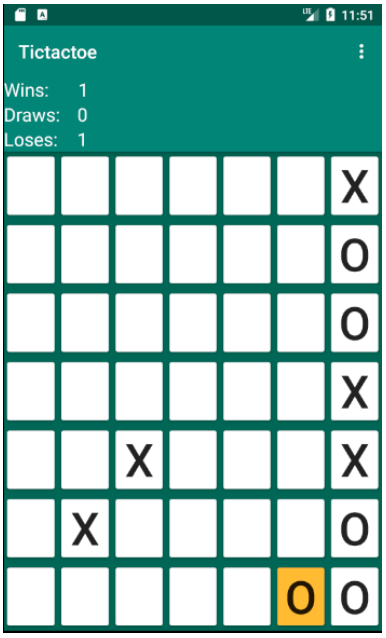
Obrázek 2: Vybrání velikosti hracího pole

Po výběru velikosti mapy aplikace vygeneruje hrací pole o zvolené velikosti a čeká, než první hráč (vy) zahájí hru prvním tahem. Příklad vygenerované mapy 7x7 lze vidět na obrázku 3.



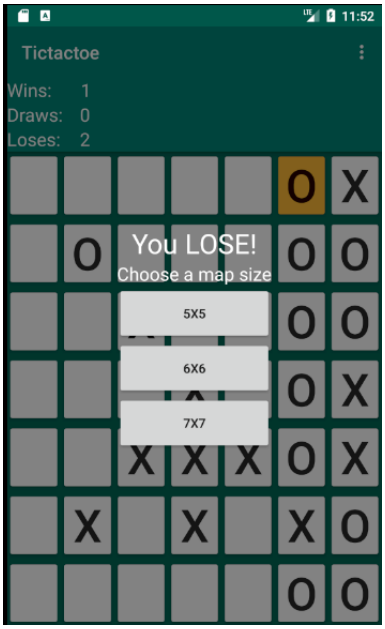
Obrázek 3: Vygenerované hrací pole

Jakmile uživatel klikne na jakékoliv pole na hrací ploše, označí se toto pole značkou **X** a už na něj nelze kliknout. Po odehrání hráčovo tahu bude hrát počítač, který podle implementovaného algoritmu vybere nějaké políčko a to označí značkou **O** Políčko zvolené počítačem bude zároveň označené žlutou barvou, aby mohl první hráč zjistit, které políčko vybral počítač. Příklad vybraného políčka počítačem lze vidět na obrázku 4.



Obrázek 4: Tah počítače

Hra končí, pokud jeden z hráčů poskládá vítěznou kombinaci (o velikosti 5) a nebo všechny políčka budou zabraná, aniž bude existovat vítězná řada, tudíž vznikne remíza. Po dohrání hry jsou aktualizovány statistiky a uživatel může začít novou hru zvolením nové velikosti hracího pole. Příklad dohrání hry do úplného konce lze vidět na obrázku 5.



Obrázek 5: Konec hry

4 Řešené problémy

Při vyvíjení aplikace jsem narazil na jeden jediný problém, a to byla časová náročnost řešících algoritmů pro AI. Jako první jsem měl naimplementován algoritmus **Minimax** a jeho rozšíření **Alpha-beta prunning**, bohužel pro hrací pole větší jak 3x3 byla časová doba pro jeden tah AI velice dlouhá a nepodařilo se mi implementovat limitaci algoritmu podle hloubky zanoření. Následně jsem použil nejjednodušší řešení, a to algoritmus **Depth-First Search**. Pro tento algoritmus byla taktéž časová doba pro tah AI víceméně stejná jako u předchozích algoritmů ale s tím rozdílem, že se mi podařilo implementoval limitaci podle hloubky zanoření. Maximální zanoření DFS může být maximálně 4 a to z důvodu přijatelné hrací doby jednoho tahu AI pro hrací pole 7x7.

5 Testování

Aplikace byla vyvíjena a testována v Android Studiu verze 3.5.3 (od firmy JetBrains) v jazyce JAVA. Při vývoji bylo využito JRE verze 1.8.0_202. Jako testovací mobil byl vybrán Pixel 2 s API verzí 24 a operačním systémem Android ve verzi 7.0 (Nougat).

6 Závěr

Vytvořená mobilní aplikace **TicTacToe**, neboli piškvorky, splňuje základní funkcionalitu stejnojmenné hry. Uživatel je schopný hru bezproblémově dohrát a ukončit ji v jakémkoliv případě. Ovládání aplikace je velice jednoduché, stačí pouze klikat na políčka v hracím poli a podle zobrazeného čísla u některých políček odhadovat, na jakém políčku se můžou nacházet bomby. Bomby jsou rozmístěny náhodně, v každé hře jsou na jiné pozici.

Při testování se nevyskytly žádné chyby, které by měli zásadní vliv na funkčnost aplikace. Testovány byly především všechny stavy, které se mohli u této hry vyskytnout, jako například výhra/remíza/prohra nebo třeba kliknutí již vybraného políčka.