Ročníkový projekt

27. dubna 2023

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Předmět: Programování

Téma: Hra bitevní lodě

Autor: *Jiří Petřík*

Třída: 2.E

Vyučující: Mgr. Jan Lána

Tř. vyučující: Mgr. Blanka Hniličková

Čestné prohlášení:

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené.

Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne ....

Poděkování

Děkuji Mgr. Janu Lánovi za vstřícnost a odborné vedení ročníkové práce.

Anotace

Cílem ročníkové práce bylo naprogramovat počítačovou verzi hry Bitevní lodě v jazyce Java. V této hře si oba hráči položí na svojí desku políčka lodí a poté se snažíte potopit nepřátelské lodě tím, že střílí protivníkovu desku. Vyhrává ten kdo první potopí všechny nepřátelské lodě.

Abstract (English)

The goal of the year project was to program the computer version of Battleships game in Java. In this game, both players place ship squares on their board and then try to sink the enemy ships by shooting the opponent's board. The first to sink all enemy ships wins.

Obsah

1 Zadání projektu **2**
1.1 Upřesnění zadání 2
1.2 Vlastní návrhy 2

Úvod **3**

2 Vývojové prostředí **3**

3 Bitevní lodě **4**
3.1 Pravidla hry 4

4 Architektura programu **5**
4.1 Grafické rozhraní 5
4.1.1 Menu a nastavení 5
4.1.2 Postavení lodí 6
4.1.3 Hledaní lodí 7
4.1.4 Konec hry a statistiky 8
4.2 Funkcionalita programu 9
4.2.1 Blokování polí kolem lodí 9
4.2.2 Ovládání průběhu 10
4.2.3 Hledání lodí 11
4.2.4 Hledání směru lodě 13

5 Závěr **15**

1 Zadání projektu

Vytvořit program, který bude simulovat hru bitevní lodě proti počítači.

1.1 Upřesnění zadání

1. Hráč si vybere počet a typ lodí, které se ve hře budou používat
2. Program vytvoří dvě pole velikosti 10x10
3. Hráč si vybere lodě a položí je na svoji desku na platné pozice
4. Poté co hráč dokončí pokládání lodí se na desku protihráče přidají lodě, které ale hráč nevidí
5. Hráč může střílet kliknutím na desku protihráče, počítač střílí na hráčovu desku pomocí daného algoritmu
6. Program určí vítěze po potopení všech lodí jednoho ze soupeřů

1.2 Vlastní návrhy

1. Statistiky - hráč uvidí počet střel a jiné údaje o dané hře
2. Před připravené populární herní módy a mód vlastního výběru lodí
3. Kontrola a ukazatel při pokládání lodí
4. Možnost výběru oponentovy strategie
5. Ukazatel zbývajících nepotopených lodí
6. Možnost náhodně postavit lodě nebo je vzít zpět z desky

Úvod

Úkol mého ročníkového projektu je tvorba počítačové simulace hry bitevní lodě (ang. Battleships).

Téma této ročníkové práce jsem si vybral, protože jsem dříve hrával hru bitevní lodě a chtěl jsem ji zkusit zpracovat v softwarovém podání. Jelikož ale na tuto hru jsou zapotřebí dva hráči, a ne vždy je někdo druhý k dispozici rozhodl jsem se vytvořit oponenta, který je ovládán algoritmem. Do ročníkové práce jsem chtěl také přidat pár svých nápadů, které nejsou na papíře možné.

2 Vývojové prostředí

Program jsem programoval v objektově zaměřeném jazyce Java, s využitím grafického rozšíření JavaFX. Pro tvorbu vzhledu aplikace a uživatelského rozhraní jsem použil program Scene Builder. Samotný program jsem psal ve vývojovém prostředí IntelliJ IDEA.

3 Bitevní lodě

Hra Lodě byla vymyšlena ve Francii během 1. světové války. První komerční verze hry byla Salvo, publikovaná v roce 1931 ve Spojeném království společnosti Starex. V roce 1967 představil Milton Bradley verzi hry, která používala plastové desky a kolíky. Později od roku 1979 vznikali různé verze této hry na počítač.[1]

3.1 Pravidla hry

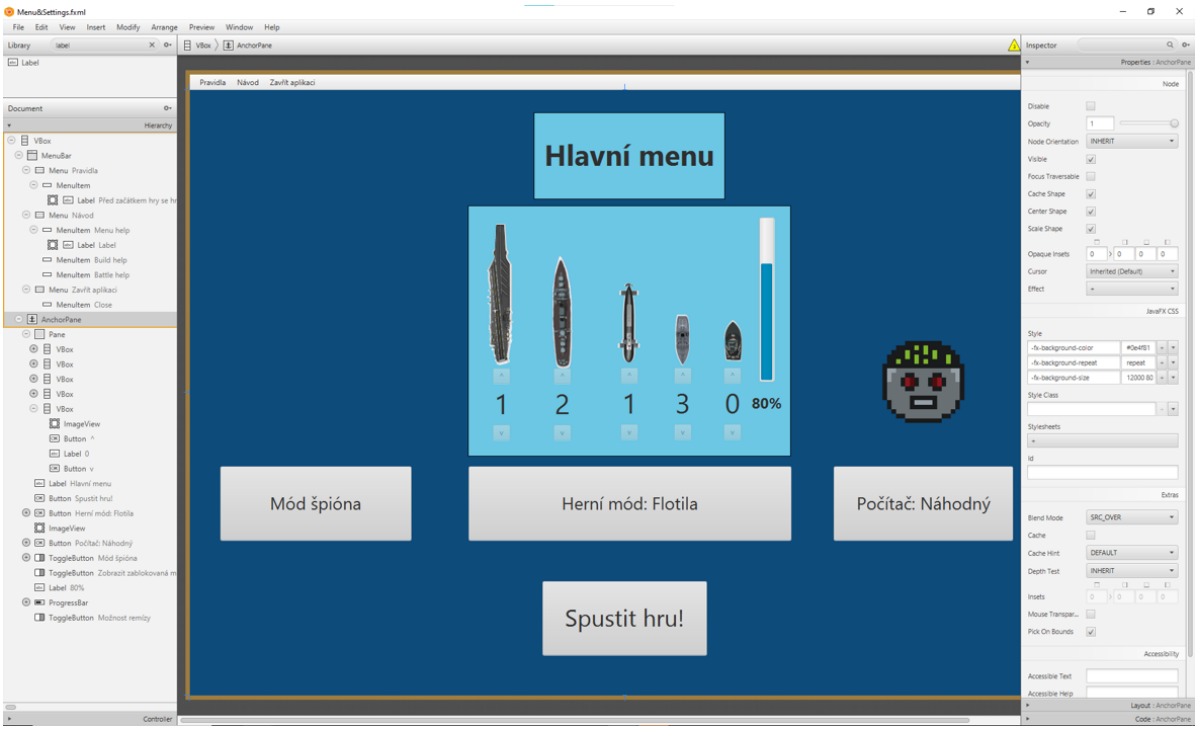
Před začátkem hry se hráči dohodnou na počtu a tvaru lodí, které do plánku budou zakreslovat. Cílem hry je nalézt a potopit všechny lodě soupeře jako první. Hráč zakreslí svoje lodě (nesmí se dotýkat stranou, ale lze je i otočit). Protivník učiní totéž do své mřížky. Začínající hráč ohlásí zvolený čtverec, kam namířil. Protivník odpoví "voda" (šedý křížek), pokud ve své mřížce na této pozici nemá žádnou loď. V opačném případě hlásí "zásah" (červený křížek). Hráč si může tečkou označovat do zmenšené mřížky radaru místa, kam již mířil, aby si udržel přehled. Zásahy lze označovat křížkem. Pokud je zasažen poslední čtvereček lodě, je plavidlo potopeno a hráč musí kromě zásahu ohlásit protivníkovi také název lodi a skutečnost, že se celá potopila (změna červených křížků na černé). Protivníkovo míření se zakresluje do velké mřížky s vlastními loděmi a jejich zásahy se značí přeškrtnutím políčka. Hráči se střídají, dokud jeden z nich nemá potopené všechny lodě. [2](upraveno)

4 Architektura programu

4.1 Grafické rozhraní

Tato část dokumentace je zaměřená na přiblížení programu uživateli a vysvětlení základních funkcí programu. Grafické rozhraní vylo vytvořeno v programu Scene Builder. Program si můžete zdarma stáhnout na stránkách zde.

4.1.1 Menu a nastavení

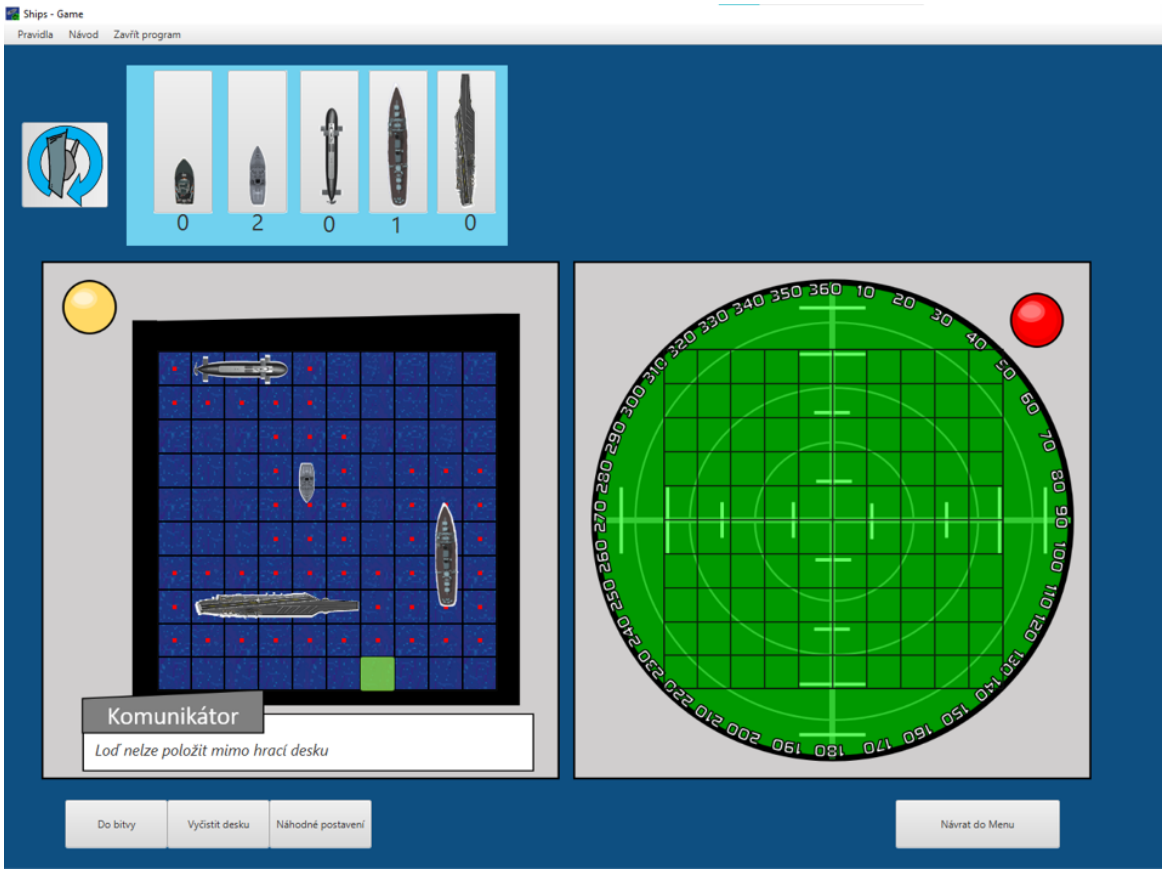


Obrázek 1: Snímek z menu v aplikaci Scene Builder

Výše na obr.1 můžete vidět hlavní menu, zde jsou tlačítka k ovládání a přípravě hry. Tlačítko počítač po kliknutí změní strategii (algoritmus) oponenta. "Mód špiona", jenž povoluje použití pomůcky užitečné zvláště pro vytváření aplikace a hledání chyb v programu. Herní mód mění počet lodí použité ve hře, mód "vlastní"umožňuje zadat vlastní počet lodí, počet je kontrolován plností desky, která je znázorněna procenty a grafem. Spuštění hry spustí hru se zadaným nastavením.

4.1.2 Postavení lodí

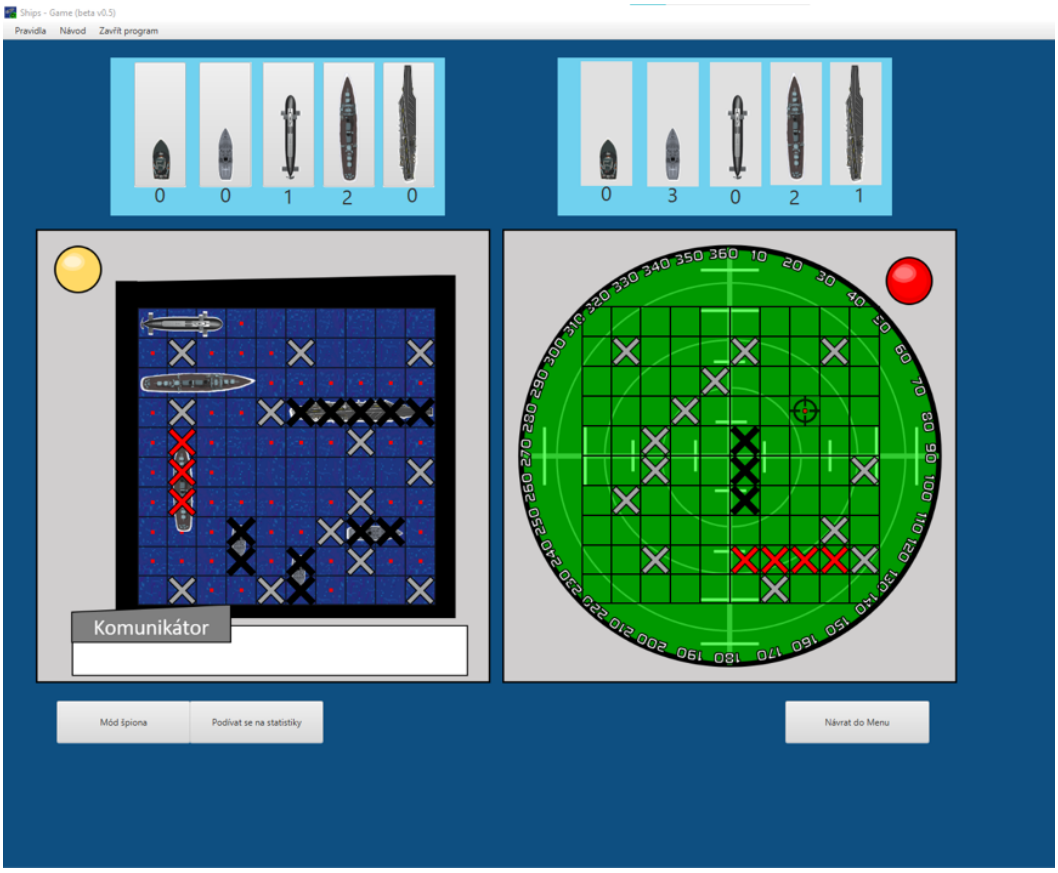
Ukázka z první fáze hry - sestavení lodí. Hráč si v horní části obrazovky vybírá velikosti lodí, pod obrázky daných plavidel je počet zbývajících lodí, které ještě musí položit. Na levou od výběru lodí je tlačítko, které mění orientaci lodí, mezi horizontální a vertikální pozicí. Vybraní lodě a orientace múza hráč položit loď na desku pomocí kliknutí na jedno z políček stojí (modré) desky. Po vybraní pole se loď vždy položí v právo nebo dolů vzhledem k vybranému bodů a otočení. Pokud by se celá loď nevešla do pole nebo by byla moc blízko lodě druhé, loď se nepoloží s v "komunikátoru"se objeví zpráva informující hráče o problému. Tlačítko "náhodné postavení "rozmístí nepoložené lodě náhodné na desku. Tlačítko "vyčistit desku"smaže všechny lodě z desky.



Obrázek 2: Ukázka hry lodě - první fáze

4.1.3 Hledaní lodí

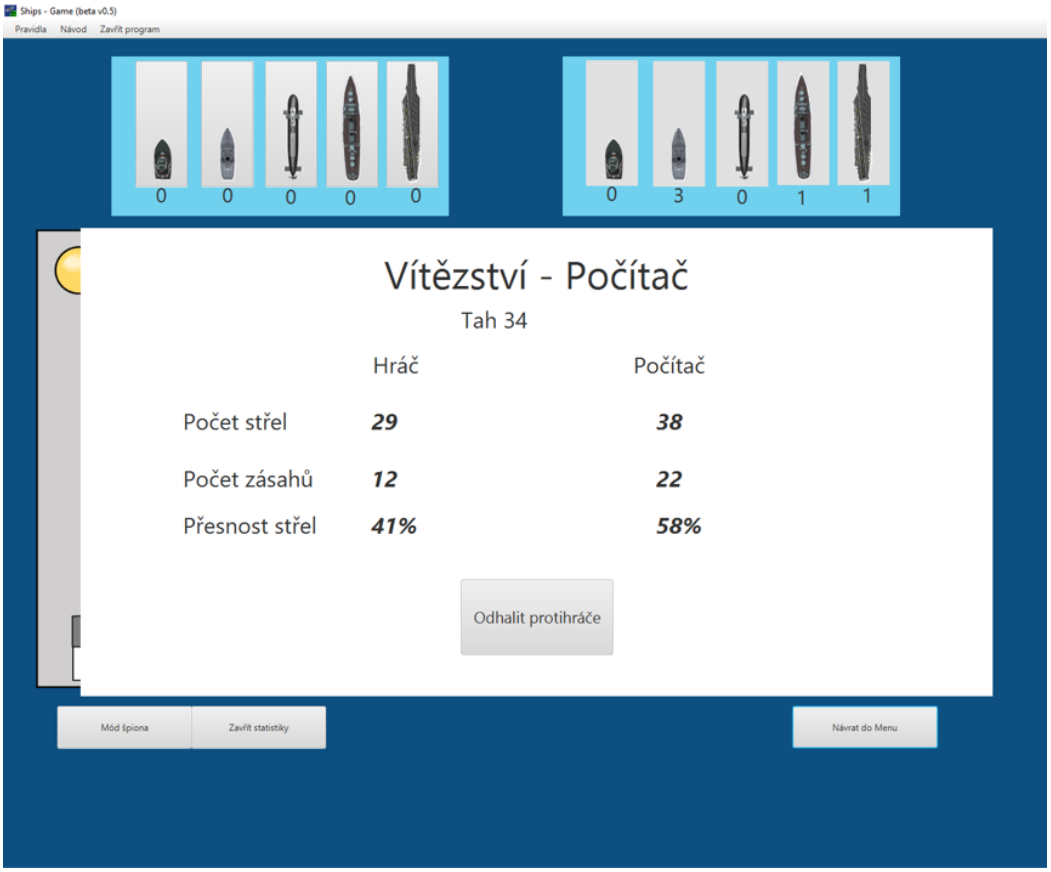
Druhá fáze hry začíná po kliknutí na tlačítko do bitvy. Nepotřebná tlačítka např. „vyčistit desku", nebo "Otočit"zmizí. V horní části aplikace se nyní místo nepostavených lodí ukazují lodě, které ještě počítač nepotopil a nad deskou oponenta se ukazují lodě nepotopené hráčem. Deska, na které hráč postavil lodě se zablokuje, takže na ni nelze kliknout a deska oponenta po přesunutí kurzoru nad její políčka ukáže ikonku zaměřovače. Hráč se smaží klikáním na zelenou desku odhalit a potopit nepřátelské lodě. Po kliknutí se políčko změní na buď to šedý kříže – hráč se netrefil, červený křížek – hráč se trefil, anebo na černý kříž, když potopil celou loď, přičemž se přebarví na černou i křížky dané lodě zasažené v minulých tazích. Po každém tahu, který hráč udělá, vystřelí také počítač na desku s loděmi hráče podle zvoleného algoritmu. Pokud se hráč nebo počítač trefí do lodě, mohou vystřelit ještě jednou. Pokud byl aktivován mód špiona, po zmáčknutí příslušného tlačítka se na desce oponenta objeví zelené tečky na každém políčku, kde má položenou loď. V průběhu hry se může hráč kdykoliv podívat na statistiky.



Obrázek 3: Ukázka hry lodě - druhá fáze

4.1.4 Konec hry a statistiky

Po sestřelením všech lodí jednoho ze soupeřů se hra ukončí a objeví se obrazovka se statistikami. Na ní lze vidět kdo vyhrál, ve kterém tahu, daný počet střel, zásahů a z nich spočítaná procentuální přesnost střely pro hráče i počítač. Pod hodnotami statistik se nachází tlačítko, které funguje stejně jako tlačítko "mód špiona", akorát tentokrát slouží už jen pro odhalení zbývajících lodí počítače, pokud chce hráč zjistit kam měl střílet.



Obrázek 4: Ukázka statistik po dokončení hry - třetí fáze

4.2 Funkcionalita programu

V této části se vám pokusím vysvětlit některé části kódu, které mi přišli zajímavé.

4.2.1 Blokování polí kolem lodí

Tato funkce je součástí algoritmu pro postavení lodě a zabezpečuje vytvoření červených teček kolem lodí, které označují místa, na která se již žádná loď nesmí postavit.

Funkce zjistí, jestli je daná loď otočená pomocí booleanu rotated a délky lodi zaznamenané v int shipLength vytvoří kolem lodi jakýsi obdélník z políček, jehož souřadnice zapíše do ArrayListu tempBlocked, který funguje jako dočasné úložiště zablokovaných polí, které se používá jak u hráč, tak u počítače. Funkce také tvoří červené čtverečky na zablokovaných místech, ale pokud pomocí booleanu AIon detekuje, že loď postavil počítač, tak čtverečky na desku nepřidá, aby neodhalila pozice jeho lodí. (1. fáze, ShipBuilder.java)

```
940 /*Vytvoreni zablokovanych poli*/
941     for (int g = 0; g < 3; g++) {
942         for (int j = 0; j < (shipLength + 2); j++)
943         {
944             /*blokace policka mimo desku*/
945             if (!rotated) {
946                     int temp = g;
947                     g = j;
948                     j = temp;
949              }
950                 if ((current.x - 1 + j) <= row - 1 &&
951                 (current.x - 1 + j) >= 0 &&
952                 (current.y - 1 + g) <= row - 1 &&
953                 (current.y - 1 + g) >= 0)
954                 {
955                     tempBlockedList.add(new coordinates
956                      (current.x - 1 + j, current.y - 1 + g));
957                     if (!AIon) {
958                         /*indikator blokace*/
959                         Rectangle rect = new Rectangle(5, 5);
960                         rect.setFill(Color.RED);
961                         GridPane.setHalignment(rect, HPos.CENTER);
962                         GridPane.setValignment(rect, VPos.CENTER);
963                         tempGrid.add
964                             (rect, current.x - 1 + j, current.y - 1 + g);
965                     }
966                  }
967
968              if (!rotated) {
969                 int temp = j;
970                 j = g;
971                 g = temp;
972             }
973         }
974     }
```

4.2.2 Ovládání průběhu

```
940 void BattleController() {
941
942         ShootShip();    //PLAYER TURN
943         if (tryAgain) {
944             return;
945         }
946
947         if (result != 0) {    //WIN CHECK
948             enemyGrid.setDisable(true);
949             ToStats();
950             return;
951         }
952
953         AIon = true;     //AI TURN
954         tryAgain = true;
955         while (tryAgain) {
956             SwitchTemp(false);
957             BotAlgorithm.BotAttack();
958             SwitchTemp(true);
959             ShootShip();
960         }
961
962         AIon = false;  //SET TO DEFAULT
963
964         if (result != 0) {    //WIN CHECK
965             ToStats();
966             enemyGrid.setDisable(true);
967         }
968     }
```

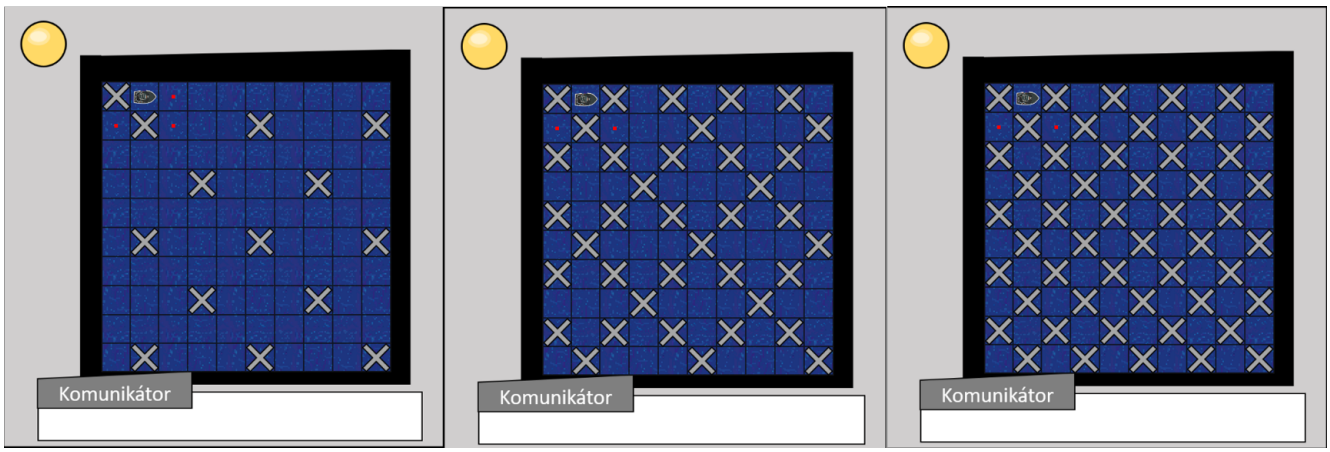
Pomocí této funkce funguje celý chod střídání se hráčů ve střílení, booleanová proměná tryAgain má dva účely, ten první je když je zasaženo nějaké pole znovu, druhé využití této funkce spočívá v tom, že když se trefíte do nějaké lodě střílíte znovu.

Průběh: Hráč vybere pozici střelby, pokud bylo dané pole již zasaženo, nebo se trefil do lodě soupeře, tak střílí do té doby, než zasáhne nové pole, které které nebylo zasaženo, nebo se na něm nenachází loď počítače. Po konci tahu hráče se zkontroluje, jestli v tomto tahu hráč nezvítězil a zapíší se nové hodnoty (střely, zásahy) do statistik. Pokračuje tah počítače, ten si první přehodí všechny hodnoty pomocí metody SwitchTemp na objekty (ArrayListy, Booleany, Inty) ke kterým má přístup Algoritmus, který vybere pozici, na kterou má počítač vystřelit metodou ShootShip(). Posledně se přehodí AIon základní hodnotu – false, a udělá se poslední kontrola vítěze a aktualizace statistik. Tento průběh se opakuje, dokud jeden ze soupeřů nevyhraje, nebo se dokud program se program nevypne.

(2. fáze, ShipsCotnroller.java)

4.2.3 Hledání lodí

Pro hledání lodí počítačem v módu "Stratég"algoritmus využívá matematických operací, aby vytvořil list cílů, které poté náhodně zasahuje. Po střele na všechny souřadnice desky se fáze zvětší a list se naplní novými cíli. Po 3 fázi je list doplněn o zbytek nezasažených políček. Celá metoda využívá číselné proměnné xOffset a yOffset, která posouvá celý seznam cílů o 0 nebo 1 do daných stran. Jak vypadají graficky fáze se můžete podívat na obr .5. (2. fáze hry, BotAlgorithm.java)

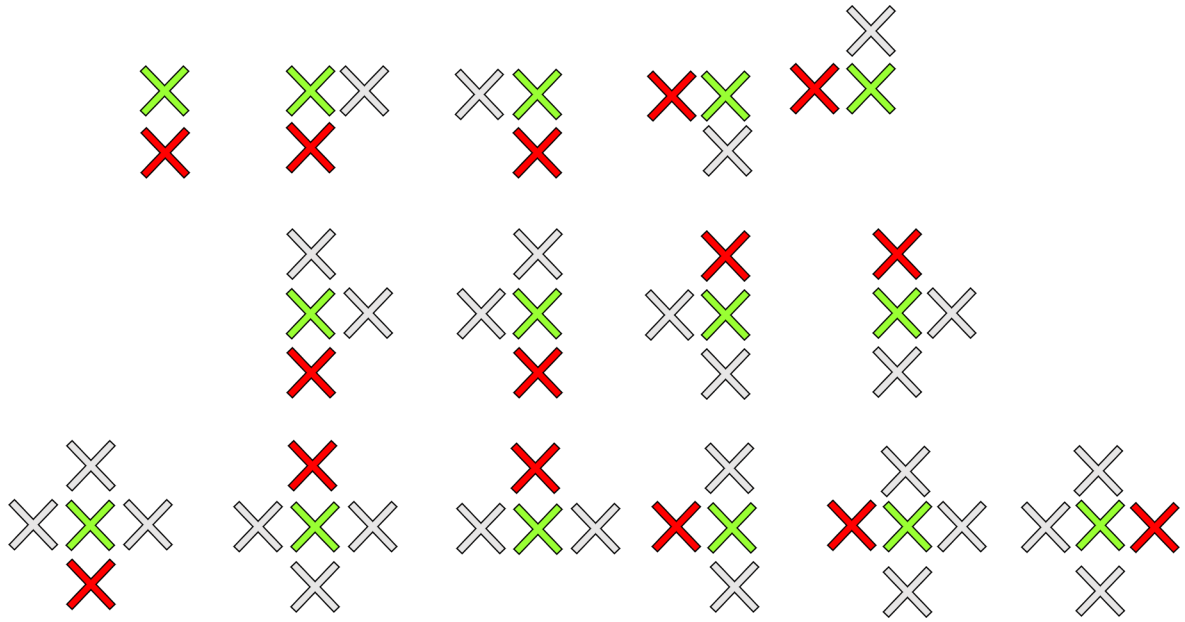


Obrázek 5: Ukázka algoritmu pro hledání lodí

```
78      if (faze == 1 && targets.size() == 0) { //Faze 1;
79           for (int i = 0; i < 5; i++) {
80              for (int j = 0; j < 3; j++) {
81                  if (i % 2 == 0) {
82                      targets.add(new coordinates
83                      (i * 2 + xOffset, j * 4 + yOffset));
84                  }
85                  if (i % 2 != 0 && j * 4 + 2 + yOffset < 10)
86                      {targets.add(new coordinates
87                      (i * 2 + xOffset, j * 4 + 2 + yOffset));
88                    }
89              }
90         }
91      }
92
93      if (faze == 2 && targets.size() == 0) { //Faze 2
94          for (int i = 0; i < 5; i++) {
95              for (int j = 0; j < 5; j++) {
96                targets.add(new coordinates
97                (i * 2 + (1 - xOffset), j * 2 + (1 - yOffset)));
98               }
99           }
100     }
101
102     if (faze == 3 && targets.size() == 0) {  //Faze 3
103         for (int i = 0; i < 5; i++) {
104             for (int j = 0; j < 3; j++) {
105
106                 if (i % 2 == 0 && j * 4 + 2 + yOffset < 10) {
107                         targets.add(new coordinates
108                         (i * 2 + xOffset, j * 4 + 2 + yOffset));
109                 }
110                  if (i % 2 != 0)
111                      targets.add(new coordinates
112                      (i * 2 + xOffset, j * 4 + yOffset));
113             }
114          }
115     }
```

4.2.4 Hledání směru lodě

Tato funkce se spustí vždy, když počítač zasáhne novou loď. První fáze kódu se snaží zjistit, jak vypadá okolí zasaženého bodu (kolik je vedle již zasažených polí). Po zjištění okolí postupuje algoritmus následovně ve střílení do jeho sousedních polí. Pole mimo desku se také počítá jako zablokované pole. (Zelený křížek je právě trefená loď, šedý křížek znamená, že toto pole bylo již zasaženo, ale loď se na něm nenachází, červené pole určuje, kam algoritmus vystřelí.) Tento algoritmus se opakuje, dokud nebude zasaženo druhé pole lodě, které pak určí orientaci lodě. (3. fáze, BotAlgorithm.java)



Obrázek 6: Ukázka algoritmu pro hledání směru lodě

```
/*vpravo*/  {
            if (currentCoordinates.x + 1 > 9 ||
                (tempBlockedList.get(i).x == currentCoordinates.x + 1 &&
                tempBlockedList.get(i).y == currentCoordinates.y))
                    {
                    right = true;
                    }

/*vlevo*/   if (currentCoordinates.x - 1 < 0 ||
                (tempBlockedList.get(i).x == currentCoordinates.x - 1 &&
                tempBlockedList.get(i).y == currentCoordinates.y))
                    {
                    left = true;
                    }

/*nahoru*/  if (currentCoordinates.y - 1 < 0 ||
                (tempBlockedList.get(i).x == currentCoordinates.x &&
                tempBlockedList.get(i).y == currentCoordinates.y - 1))
                    {
                    up = true
                    }

/*dolu*/    if (currentCoordinates.y + 1 > 9 ||
                   (tempBlockedList.get(i).x == currentCoordinates.x &&
                   tempBlockedList.get(i).y == currentCoordinates.y + 1))
                     {
                     down = true;
                     }
            }

 /*smer*/   if (left && right) {
                if (up) {currentCoordinates.set(Pivot.x, Pivot.y + 1);
                }
                else {currentCoordinates.set(Pivot.x, Pivot.y - 1);
                }
                return;}

            if (up && down) {
                if (left) {currentCoordinates.set(Pivot.x + 1, Pivot.y);
                }
                else {currentCoordinates.set(Pivot.x - 1, Pivot.y);
                }
                return;}

            if ((up && right) || (up && left)) {
                currentCoordinates.set(Pivot.x, Pivot.y + 1); return;}
            if ((down && right) || (down && left)) {
                currentCoordinates.set(Pivot.x, Pivot.y - 1); return;
                }

            if (down || up)
                {currentCoordinates.set(Pivot.x - 1, Pivot.y);
            }
            else {currentCoordinates.set(Pivot.x, Pivot.y + 1);
            }
```

5 Závěr

Vytvořil jsem aplikaci, která simuluje hru bitevní lodě, do které jsem implementovat pár inovací a návrhů.

Své cíle jsem splnil a povedlo se mi přidat i nějaké funkce navíc.

Kdybych měl na projekt více času, určitě bych se věnoval více vizuální stránce a vytvořil příjemnější uživatelské rozhraní. Dále jsem přemýšlel o přidání více tvarů lodí, což mi ale stávající struktura programu a vytvořené algoritmy nedovolují. Ročníkový projekt mě bavil a myslím si, že se mě programátorské schopnosti velice posunuly od minulé ročníkové práce.

Reference

[1] Wikipedia, "Lodě," 17. 4. 2023. Stránka: wikipedia.org, https://cs.wikipedia.org/wiki/Lod%C4%9B.

[2] ReklamniKarty, "Námořní bitva - pravidla hry," 2008. Stránka: www.ReklamniKarty.cz , http://www.reklamnikarty.cz/pravidla_lode.htm.

Použité zdroje při programování:

- Add buttons to gridpane
  https://stackoverflow.com/questions/24715911/javafx-adding-button-to-grid-pane
- Button hover CSS style
  https://stackoverflow.com/questions/30680570/javafx-button-border-and-hover
- Button generator
  https://css-tricks.com/examples/ButtonMaker/
- Background image
  https://stackoverflow.com/questions/23515172/set-background-image-the-same-size-as-the-window-screen-in-java-app
- Gridpane background image
  https://stackoverflow.com/questions/33973830/javafx-gridpane-background-image
- Switching scenes
  https://www.youtube.com/watch?v=hcM-R-YOKkQt=108s
- Button check
  https://stackoverflow.com/questions/51456418/how-to-check-which-button-in-fxml-has-invoked-a-function-in-controller-java-wh
- Imageview set image
  https://stackoverflow.com/questions/9738146/javafx-how-to-set-scene-background-image