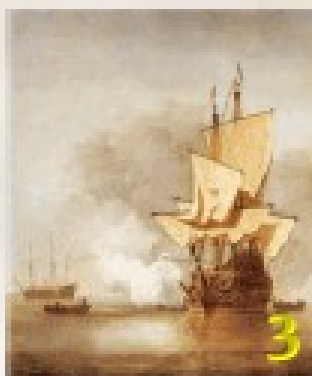


# RYCHLO LODE

*X36ALG - semestrální práce 2008/09*



*Matěj Šimek*

*matej.simek@gmail.com*

## Požadavky

- Práce musí splňovat dané zadání a bezchybně fungovat.
- Zdrojový text práce musí být přiměřeně obsáhlý (např. 250 řádek zdrojového kódu). Pokud zadání práce připouští naprogramovat řešení kratším způsobem, musí student obohatit práci o další funkcionalitu např. nápovědu, archivaci atp.
- Student musí prokázat, že práce je jeho samostatným dílem. Tuto podmínku nelze prakticky prokázat jinak, než že student demonstruje před cvičícím podrobnou znalost zdrojového textu práce. Student musí být např. na požádání schopen udělat v práci drobnou změnu.
- Práce musí být správně zformátována a přiměřeně zdokumentována. V každém případě by součástí zdrojového textu mělo být informace o autorovi práce, název, stručné zadání, podmínky k provozování atd. Vlastní program by měl mít komentovány významy globálních proměnných a procedur. Na zvážení je, zda student musí odevzdat samostatnou technickou dokumentaci práce.

## Úvod

Jako téma semestrální práce jsem si vybral hru Lodě. Shodou okolností jsem před 5 lety programoval pro tehdejší herní vývojářský server GameCode.cz stejný projekt do soutěže, obsadil jsem dokonce přední příčku (3.). V té době jsem na základní škole prozkoumával možnosti programování obecně a vlastnosti jazyka Visual Basic (5), o kterém jsem měl doma spoustu knih. Jak ale každé začátky bývají, zdrojový kód byl velmi nepřehledný, chaotický a slovíčko objektové programování nepatřilo do mé báze znalostí.

## Cíl

*Za pomoci všech znalostí přenést původní koncept do prostředí Javy s použitím objektově orientovaného programování. Jako zjednodušení použít původní grafické podklady a uživatelské rozhraní.*

## Realizace

Na obrázku vpravo je vidět screenshot původního programu. Této koncepce jsem se držel i nadále.

Odshora můžeme vidět obrázkové logo, níže samotnou hrací plochu s polem 7x7 políček a ovládací tlačítka pro spuštění a ukončení hry.

Jako uživatelský vstup je v tomto případně brána pouze myš. Přístupnost je tedy na nulové úrovni a uživatel jakkoliv vybočující z průměru, s mentální, tělesnou či zrakovou dysfunkcí si tuto hru neužije.





V původním programovém zpracování se vůbec nepoužívala metoda pro vykreslení grafiky a všechny prvky (pozadí, jednotlivá políčka, mřížka složená z 12 čar apod.) byly natvrdo umístěny a vytvořeny v tvůrci uživatelského prostředí. I jednotlivé stavy se odvozovali od jejich vlastností a obsahu toho či onoho obrázku.



Vyloučeny byly tedy jakékoliv úpravy při programování, běhu programu a hlavně samotné vytvoření bylo velmi náročné.

V novém návrhu jsem tedy samozřejmě už od začátku nepočítal s tímto způsobem realizace. Nyní důsledně odděluji abstraktní část od části starající se pouze o zobrazení.

Základním stavením prvek hracího systému je samotné políčko. Může nést 4 různé stavy – prázdné = neobjevené, obsazené = loď, potopené = ropná skvrna, která zůstala po plavidlu a voda = prohledané území s pouhou mořskou hladinou. Grafické vyobrazení můžete vidět níže.

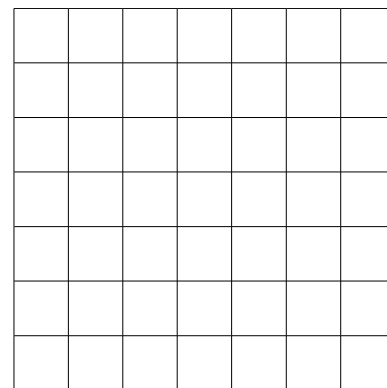
-  Otazník značí neobjevené pole (0)
-  Základní bitevní plavidlo (1)
-  Ropná skvrna (2)
-  Mořská hladina (3)

### Sez. Grafické prvky

V kódu je políčko interpretováno jako třída `Policko` a svůj obsah si nese v jedné proměnné stav. Ta může nabývat hodnot 0–3 (viz předešlý seznam). Metody pro obsluhu: `NastavPrazdne()`, `NastavObsazene()`, `NastavPotopene()`, `NastavProhledane()`, `NastavNahodne()`, `VypisObsah()`, jejíž funkce je jasná z názvu, a metoda `Strelba()`, která se stará a obsluhu střelby na dané pole – zda jsme něco zasáhli, potopení lodi, odhalení hladiny apod.

O poskládání políček do jednoho celku se stará třída `HraciPole`. Ta představuje už celou herní plochu o zadaných rozměrech, konkrétním obsahu a počtu lodí. Obsah je uskladněn v jednorozměrném poli `Políček`. Představit si ho můžeme jako tabulku (viz. obr vpravo).

Pomocí vícenásobného konstrukturu můžeme vytvořit vzorové náhodné pole (pro testovací účely); „chaotické“ pole, sloužící pouze pro demo na úvodní obrazovce a normální pole pro herní účely, u kterého volíme v tomto pořadí: náhodné zaplnění lodí (pro počítač), šířku, výšku hracího plánu a celkový počet lodí pro jednotlivé účastníky.



Obr. *HraciPole*

Další implementovanou metodou je `Strelba()`, která s v této vrstvě už stará o počet lodí v poli při zásahu, avšak nezasahuje do samotného obsahu políčka. Dále `StavPolicka()` navracející obsah buňky, `PocetLodi()` navracející zbývajících počet plavidel a pro účely testování `toString()` vypisující obsah plochy do řetězce.

Průběh hry zajišťuje třída `Hra`. Opět jsme postoupili o pomyslný stupínek výše a nestaráme se nyní toliko o obsah, jako o průběh. Při tvorbě hry si můžeme zvolit ze 4 druhů – hra uživatele, hra počítače, tzv. demo a „stínová hra“. První 3 typy jsou jasné, objasním ten poslední. Jelikož se jedná z principu o hru pro dva hráče, nechceme aby si oba navzájem viděli polohu svých lodí, zavedl jsem tento typ pro jakési maskování. Hráč vidí pouze horní stínovou vrstvu, kterou si postupně odkrývá

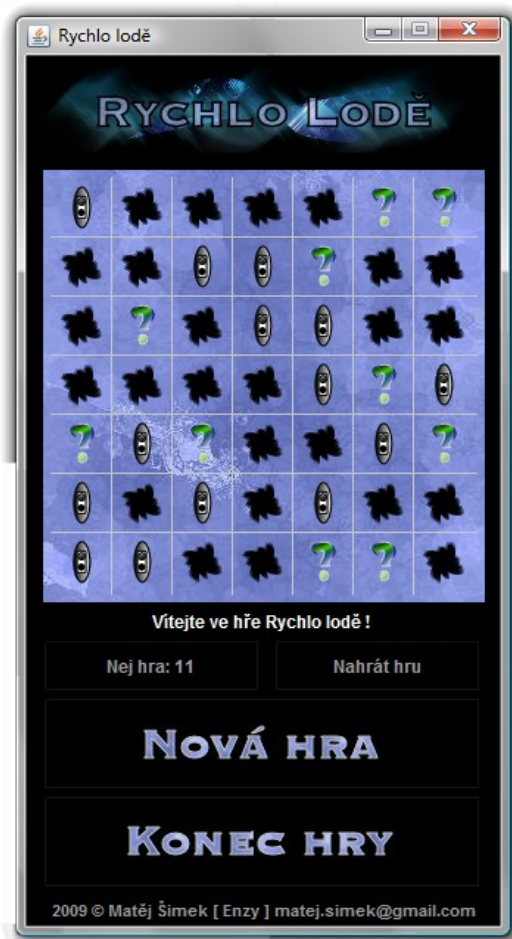
svými tahy a ta pravá nesoucí informace o protihráči je schována pod ní. Střelba je však zachytávána pouze vrstvou spodní.

Metoda `Strelba()` se zde stará o počet lodí ve hře vzhledem k maximu (0 lodí značí výhru), počet tahů hráče, přehrání zvukových efektů, průstřelnost stínové vrstvy a její odkrývání. `StavPolicka()` navrácí obsah políčka na zadané pozici, `isRunning()` zda je daná hra aktivní, `Ukonceni()` se stará o akce související se závěrem hry – určení vítěze, zvukový efekt a zavolání zpracování dosaženého score. `UlozeniHighScore()` vyhodnocuje výkon vzhledem k předešlým a zapisuje tyto hodnoty do souboru.

Metody `UlozeniHry()`, `NacteniHry()` se starají o uložení / načtení rozehrané hry na disk do souboru `savegame.bin`. Fungují na podobném principu jako serializace.

Zatím jsme používali pouze abstraktní data uložená někde v paměti. O jejich zobrazení se stará třída `JBHraciPole`, což je rozšířené klasické tlačítko `JButton` z grafické knihovny Swing. Metoda `paintComponent()` je ale přepsána tak, aby nám vykreslovala hrací plán, konkrétně grafické prvky ([viz. Sez. Grafické prvky](#)) uspořádané do mřížky, na obrazovku. Dále se stará o zpracování uživatelského vstupu v podobě kliknutí na objekt v metodě `mysKlik()`, kde se souřadnice  $x, y$  převedou na index políčka a ten se pak odešle přidružené hře pro střelbu.

Ještě bych chtěl uvést dvě použité třídy. `ImagePanel` rozšiřuje `JPanel` a pomocí něho je vykresleno pozadí hrací plochy – vstupní obrázek se vykresluje opakovaně, aby zaplnil celou oblast, což je princip tzv. dlaždic. `Mrizka` nám slouží pro vykreslování rastru nad hrací plochou pro snazší orientaci.



Celé by to nemohlo fungovat bez nějakého prostoru, kde bychom zobrazili grafickou interpretaci dat a odchytili uživatelský vstup. Jedná se o `JFrame` formulář hlavní okno (z grafické knihovny Swing). Do něho program přechází automaticky z metody `main()` třídy `Main`. Dalo by se říci, že v tomto objektu probíhá nejvíce řízení programu a je tedy pro něj nepostradatelný.

Při vytvoření objektu se inicializují jednotlivé komponenty vytvořené v průvodci tvorby GUI ve vývojovém prostředí NetBeans 6.5, deklarují se objekty pro pozdější použití (obrázky, zvuky, hry), naplní se, vloží se vlastní komponenty pro vykreslení grafiky (hrací plochy) a zavedou se služby uživatelského vstupu. Zobrazí se vlastní okno a hrací pole naplní demo hrou.

Výslednou startovací obrazovku můžete vidět nalevo.

Převod se tedy povedl bez větších viditelných změn, alespoň co se úvodní obrazovky týče.

## Vyhodnocení

Základní myšlenku se mi podařilo splnit, program je nyní převeden do Javy a objektově orientovaný. Z pohledu uživatele se jedná pouze o drobné detaily, které se dají doladit.

Při ruční tvorbě GUI v NetBeans jsem se setkal s trochu nepřátelským chováním, způsobené automatickým kotvením objektů a nekorektní úpravou velikostí a pozice okolních objektů. V tomto mi přišel před 5 lety způsob Visual Basicu příjemnější. Nic si však nezadá s komfortní programovou tvorbou komponent Swingu v Javě.

Nynější stádium programu je ve stavu předváděcí demoverze. Na implementaci volby vlastních lodí, hry s druhým člověkem na jednom počítači či přes internet nezbyl bohužel čas. I tak se jedná o použitelnou mini-klikačku na krátké nudné přestávky, když už klasické Miny přestanou bavit.

Pokud bych se vývojem zabýval dál, pustil bych se na 3D prostředí, které je mnohem zajímavější.

19.01.2008

Vypracoval Matěj Šimek

[matej.simek@gmail.com](mailto:matej.simek@gmail.com)

SVN repositář: <http://code.google.com/p/matejsimek-alg1-semestralni prace/>

## Použité zdroje

Osobně děkuji panu Ing. Balíkovi za jeho příklady, které mi pomohli s porozuměním tvorby a fungování GUI.

- (1) Základy grafiky v Javě, Ing. Miroslav Balík Ph.D. [online], poslední revize z 19.1.2009.  
Dostupné z: <<http://service.felk.cvut.cz/courses/X36ALG/ruzne/grafika.html>>
- (2) Java: Přehrávání zvuku, Pavel Novák [online], poslední revize z 19.1.2009.  
Dostupné z: <<http://pavel-novak.net/clanky/java-prehravame-zvuk.html>>
- (3) seriál Java, Linuxsoft.cz [online], poslední revize z 19.1.2009.  
Dostupné z: <[http://www.linuxsoft.cz/article\\_list.php?id\\_kategorie=192](http://www.linuxsoft.cz/article_list.php?id_kategorie=192)>



## Příloha 1: Uživatelský manuál

### Systémové požadavky

- CPU 500 MHz
- 256 MB RAM
- 40 MB na pevném disku
- rozlišení alespoň 1024x768 (32 bit)
- zvuková karta
- myš + prsty
- Java JRE 1.5

### Poučení

Pokud jsi šťastným vlastníkem alespoň jedné ruky, slabého zraku a špetkou domýšlivosti, rovnou tuto část přeskoč, neboť je určena pouze těžkým případům.


### Spuštění



Hru spustíš souborem `semestralnipracel_lode.jar`

Případně spuštěním z příkazové řádky příkazem:

```
java -jar semestralnipracel_lode.jar > debug
```

### Hlavní okno

Hra se spustí v tzv. demo módu. Pro spuštění nové hry stačí kliknout levým tlačítkem myši na tlačítko Nová Hra 

Zobrazí se ti hrací plocha plná otazníků , které představují neprozkoumané pole. Vyber si nějaký a klikni na něj levým tlačítkem myši. Pokud uslyšíš žbluňknutí, bohužel jsi se netrefil a tvá střela skončila na dně oceánu. Jiné je to ale v případě výbuchu a následné  ekologické havárie zbylé nafty, to jsi měl štěstí a potopil jsi soupeřovu loď!

Soupeř má celkem 10 lodí a ty je musíš najít všechny při co nejmenší spotřebě munice. Kolik si vystřílel můžeš vidět v políčku Počet tahů.

Nejlepší, tzn. nejnižší počet tahů se po skončení hry ukládá a tak si můžeš postupem času zlepšovat!

Až tě to přestane bavit, klikni prostě na Konec hry .

Pokud objevíš nějakou chybu, nech si jí pro sebe, protože tato hra se už dále vyvíjet nebude.