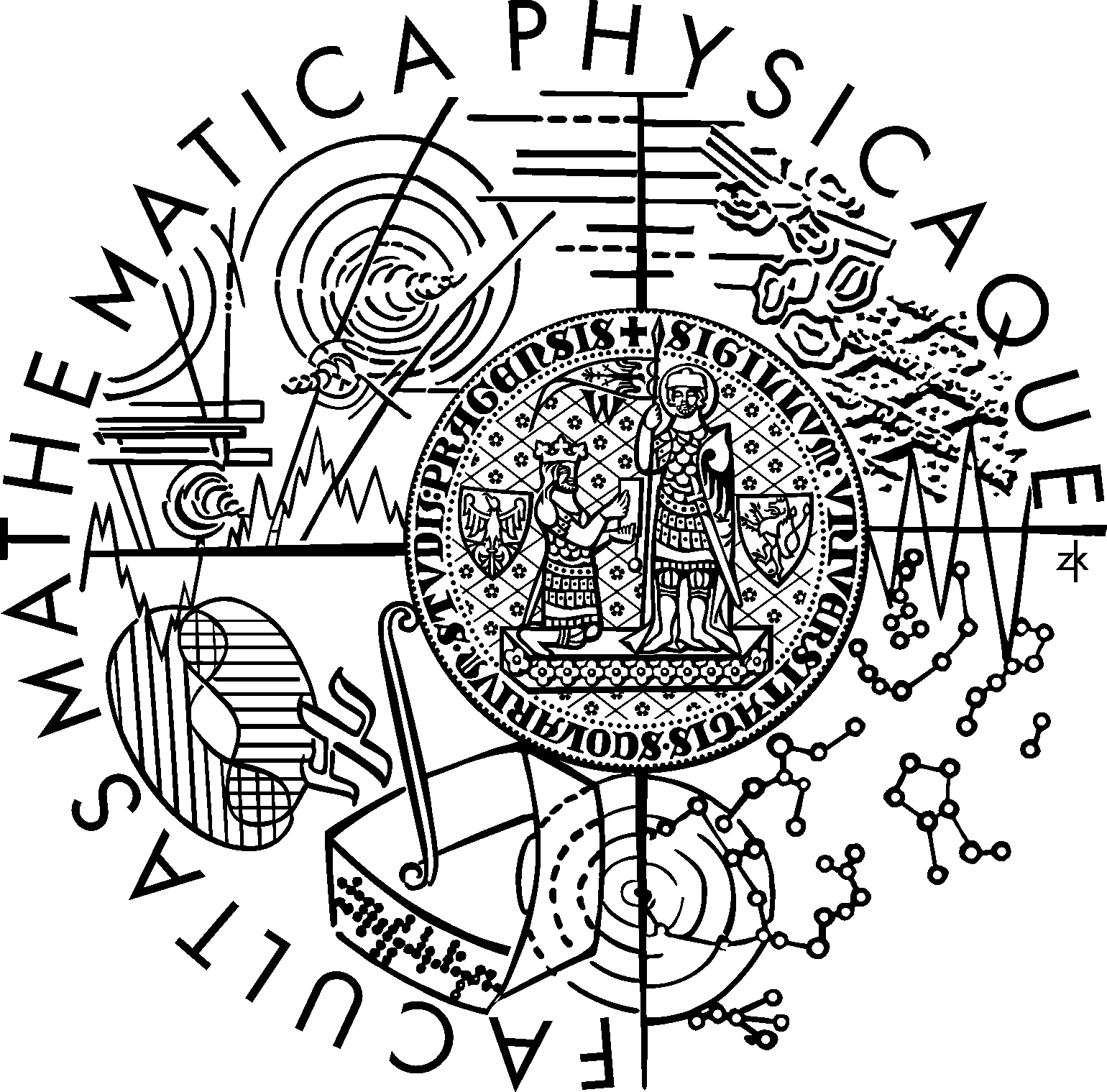
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta



Dokumentace k programu

**Brainerman**

Zápočtový program

Programování II (NPRM045), LS 2015/2016

**Kateřina Břicháčková**

Obsah

[Úvod 3](#_Toc457255270)

[Uživatelská část 4](#_Toc457255271)

[1 Spuštění hry 4](#_Toc457255272)

[2 Pravidla hry 4](#_Toc457255273)

[Programátorská část 6](#_Toc457255274)

[1 Zadávání nových levelů 6](#_Toc457255275)

[2 Timer 6](#_Toc457255276)

[3 Grafika 6](#_Toc457255277)

[4 Hlavní třídy 7](#_Toc457255278)

[4.1 Třída Monster 7](#_Toc457255279)

[4.1.1 Monster1 7](#_Toc457255280)

[4.1.2 Monster2 8](#_Toc457255281)

[4.1.3 Monster3 8](#_Toc457255282)

[4.2 Třída Map 8](#_Toc457255283)

[4.2.1 Metoda MoveObjects() 8](#_Toc457255284)

[4.2.2 Metoda moveHero() 8](#_Toc457255285)

[4.2.3 Metoda checkMonsters() 9](#_Toc457255286)

[4.2.4 Metoda checkBombs() 9](#_Toc457255287)

[5 Možné modifikace programu 9](#_Toc457255288)

# Úvod

Slavnou hru Bomberman (také známou jako Dyna Blaster) zná snad každý v mém věku. Existuje nespočet variací, ať už je to původní hra pro ZX Spectrum, hra, kterou si můžeme zahrát v telefonu s Androidem, nebo moderní videohra. Proto jsem se rozhodla udělat hru alespoň něčím odlišnou, s „matfyzáckou“ tématikou. Stačilo by ale jen změnit grafiku, udělat pár úprav v pravidlech a hra by se lehce proměnila v klasického Bombermana.

Tato dokumentace obsahuje uživatelskou část, tedy popis toho, jak s programem pracovat a jak se hra hraje, a programátorskou část, ve které se zaměřuje na hlavní použité třídy a jejich metody, důležité algoritmy, nebo způsob, jak lze hru modifikovat. Nedílnou součástí dokumentace jsou komentáře přímo v kódu.

# Uživatelská část

# Spuštění hry

Hra se spouští pomocí souboru Brainerman.exe. Ve složce s ní jsou všechny potřebné soubory (obrázky, zvuky) pro správné spuštění programu.

Po spuštění se objeví první okno, ve kterém se nachází menu. Z něj je možné spustit novou hru, pokračovat ve hře (pokud je nějaká právě rozehraná), zobrazit nápovědu se stručnými pravidly a ovládáním hry, nebo hru ukončit.

Pokud začneme novou hru, zobrazí se druhé okno, ve kterém se vykresluje samotná hra. Okno tvoří herní plán a horní lišta, na které se zobrazuje skóre získané v daném levelu, zbývající čas pro úspěšné dokončení, a životy hrdiny.

# Pravidla hry

Hra má 7 levelů, každý z nich představuje jeden semestr na Matfyzu, plus jeden bonusový (státnice). Cílem levelu je v časovém limitu zabít všechny příšery (představující školní předměty) a najít exit („hospodu”), který je ukrytý pod některou ze zničitelných zdí, a do otevřeného exitu vstoupit. V každém kole je také ukrytý jeden bonus, jejich popis se nachází v nápovědě.

Hráč má na začátku 3 životy, může je ztratit dotekem s příšerou, políčkem s ohněm z vybouchlé bomby (mozku) nebo vypršením času. Naopak získat je může pouze bonusem pro přidání životu. Pokud hráč ztratí život, ztrácí všechny dosud získané výhody z bonusů.

Rozmístění políček v levelu je náhodné, proto je zbytečnné hledat bonus či exit na tom samém místě. Typ bonusu je ale pokaždé stejný pro ten daný level, stejně tak počet a typy příšer a počet zničitelných zdí.

Za zabití různých typů příšer dostane hráš různý počet „kreditů“. Ty se po úspěšném dokončení kola zobrazí.

Poslední, sedmé kolo, neboli Státnice, má speciální pravidla. Na začátku nejsou v poli žádné příšery, ale při zničení zdi se místo ní může objevit příšera. Exit se v tomto kole neotevře, dokud nejsou zničeny všechny zdi.

# Programátorská část

# Zadávání nových levelů

V Proceduře StartGame(int Level), která patří oknu GameWindow, se podle aktuálnní úrovně zavolá konstruktor třídy Map a vytvoří konkrétní level. Pokud bychom chtěli přidat nebo upravit levely, stačí jen upravit tuto metodu. Dá se nastavit název levelu, počet a typy příšer, počet zničitelných zdí, typ bonusu, stav hrdiny (=životy, dosah ohně, počet bomb a rychlost), a jaký obrázek bude představovat hrdinu (matfyzák / matfyzačka).

# Timer

Timer, který patří GameWindow, tvoří důležitou část programu. Z něj se volají události, které se (podle stavu mapy) mají stát v každém okamžiku při startu a běhu hry a při výhře nebo prohře.

Timer vypíná a zapíná zvuky a stopky, nastavuje viditelnost komponent GameWindow, vypisuje texty pro uživatele při výhře nebo prohře. Ve stavu Running volá hýbání objektů v mapě a její vykreslování.

Co se v každém stavu děje je podrobněji popsáno v komentáři přímo v kódu.

# Grafika

Celý herní plán je v každém kroce metodou Map.Portray vykreslen na bitmapu, která patří GameWindow. Je volána metoda Invalidate() a následně override OnPaint, která vykreslí bitmapu do PictureBoxu.

Obrázky, které se vykreslují, jsou načteny při vytvoření nové mapy. Políčka jsou velká 50 pixelů, pokud bychom chtěli použít jinou velikost, změnila by se hodnota proměnné Map.TileSize. Obrázky jsou tvořené ve Photoshopu, případně jde o upravené obrázky z internetu.

# Hlavní třídy

Všechny třídy, jejich metody i proměnné jsou okomentovány přímo v programu. Zde jsou některé z nich rozepsány podrobněji a je trochu přiblíženo, jak spolu souvisí a komunikují.

## Třída Monster

Všechny typy příšer jsou odvozeny od abstraktní třídy Monster. Příšera má svou mapu, ve které se pohybuje, a umí zjistit, jestli může jít rovně, doleva, doprava nebo zpátky (pro každou možnost má jednu metodu), umí se otočit a jít rovně. Umí najít následující políčko ve svém směru.

Pohyb příšery se děje v metodě Move(), kterou volá Map.MoveMonsters(). Pokud se chce příšera pohnout z jednoho políčka na druhé, neudělá to “najednou”, v jednom kroku timeru, ale v každém druhém kroce (kvůli menší rychlosti) se posune o tolik pixelů, kolik je její rychlost. Pamatuje si, kolikrát už se posunula, a pokud už zbývá do dalšího políčka méně pixelů než je její rychlost, posune se jen o ten zbytek. Výjimkou je, pokud před ní během pohybu hrdina položí bombu. Aby přes ní nepřejela, zruší se vykonávaný pohyb vpřed a příšera se vrátí na výchozí místo.

V metodě Move() se take rozhoduje, jestli se právě příšera pohybuje z políčka na políčko, nebo jestli už tam dorazila a má udělat další krok. Pokud ano, volá se abstraktní metoda Step(), kterou už má každá příšera jinou, a tím se příšery mohou lišit. Dále se liší svou rychlostí a počtem kreditů, které jejich zabitím hrdina získá.

### Monster1

„Nejhloupější“ a nejpomalejší ze všech příšer. Jde rovně, dokud je to možné, a pokud není, náhodně se rozhodne mezi políčky, kam může pokračovat.

Má rychlost 1 a hráč za ni dostane 4 kredity.

### Monster2

Tato příšera v každém kroce rozhodne, jestli může jít doleva, doprava nebo rovně, a náhodně vybere. Pokud nemůže ani jedno, otočí se zpět.

Rychlost 2, skóre 7.

### Monster3

Má stejný algoritmus jako Monster2, ale je rychlejší a může procházet skrz dočasné zdi.

Rychlost 3, skóre 14.

## Třída Map

Map je zdaleka nejdůležitější třídou, využívá všechny ostatní třídy a má na starost průběh celé hry. Má svého hrdinu, seznam příšer, seznam aktivních bomb, seznam políček, kam vybouchly bomby, obrázky, které vykresluje; pamatuje si skóre, do dvojrozměrného pole znaků ukládá typ políček, jaké se v daném řádku a sloupci nachází...Její proměnné jsou popsané přímo v kódu.

### Metoda MoveObjects()

Metoda, díky níž se celý plán „hýbe“. Sama o sobě je jednoduchá, ale volá několik dalších metod třídy Map.

### Metoda moveHero()

Podle stisknuté klávesy zjistí, jestli lze posunout obrázek hrdiny daným směrem o tolik pixelů, kolik je jeho rychlost. Volá metodu isNotCollision(int X, int Y), která zjistí, co se nachází na políčku mapy, kam se chce hrdina přesunout. Kontroluje souřadnice obou rohů obrázku hrdiny podle toho, jakým směrem se chce hrdina pohnout (například při stisku klávesy pro pohyb dolů se zkontroluje levy a pravý dolní roh). Pro snadnější pohyb hrdiny je nastavená tolerance – vlevo a vpravo 5 pixelů, nahoře 10 a dole nic.

### Metoda checkMonsters()

Zjistí, jestli po pohybu příšer a hrdiny některá z příšer nezabila hrdinu. Podle toho, jaká příšera ho zabila, nastaví se parametr HeroDeath, který využije GameWindow při zobrazení příslušného textu.

Volá metodu heroTouches(Monster Monster), která určí, zda se obrázek hrdiny překrývá s obrázkem příšery. Je opět natavená určitá tolerance.

### Metoda checkBombs()

Zkontroluje aktivní bomby, zvýší jejich čas, jak dlouho jsou položeny, a pokkud tento čas přesáhne daný čas pro odpálení bomby (bombFireTime), zavolá metodu fire(Bomb Bomb), která bombu odpálí, a odstraní tuto bombu ze seznamu bomb.

Metoda fire vyšle „oheň“ do všech čtyř směrů a volá metodu destroy(int Row, int Col, Bitmap Bitmap), která zničí dané políčko, vytvoří na něm FireTile (políčko s ohněm, kvůli tomu, aby oheň hned nezmizel, ale ještě se zobrazoval) se zadanou bitmapou, která se má vykreslit (aby oheň tvořil kříž). Destroy také určí, zda se má oheň dále šířit, nebo se zarazil. Oheň se dále může zarazit, když už dále nedosáhne, a v tom případě se jako bitmapa předá „ukončující“ obrázek kříže.

# Možné modifikace programu

V průběhu psaní programu mě napadlo spoustu možností, jak by šlo hru vylepšit, a někdy v budoucnu bych se k ní určitě ráda vrátila a dotáhla jí do konce.

Bylo by dobré, aby měl uživatel možnost měnit nastevení – změnit klávesy pro pohyb hrdiny, zvětšit nebo zmenšit si herní okno a podobně.

Pokud by se chtěly přidávat složitější levely nebo dokonce světy (magisterské a doktorské studium? ☺ ), bylo by potřeba mít větší herní plán. To by se dalo vyřešit tak, že by se plán posouval tak, aby hrdina byl vždy veprostřed.

Bylo by zajímavé vytvořit příšeru, která by se nepohybovala náhodně, ale podle pohybů hrdiny – například tak, že by si plán pamatoval, před jakou dobou tam hrdina vstoupil, a podle toho by se příšera mohla rozhodnout. Nebo by počítala vzdálenost od hrdiny a zjišťovala, jestli se tím daným krokem přiblíží nebo oddálí.

Vzhledem k tomu, jak je program napsán, bylo by možné i poměrně snadno přidat druhého hráče a udělat například souboj o to, kdo bude mít větší skóre.