



# Uživatelská dokumentace k zápočtovému programu Space Invaders

Program spustí hru Space Invaders, která spočívá v zabíjení příšer. Hráč představuje vesmírnou loď, která se pohybuje po zobrazeném okně a střílí na příšery. Cílem hry je zabít všechny příšery ve hře a dojít do osmého levelu.

## Ovládání

Hráč ovládá vesmírnou loď  pomocí 3 šipek na klávesnici.  Klávesa doprava posune loď doprava. Klávesa doleva posune loď doleva. Klávesa nahoru vystřelí jednu střelu.

## Pravidla hry

Ve hře existují 3 typy příšer.




Tato příšera má pouze 1 život, tj. stačí, abychom ji jednou zastřelili a zemře. Sama příšera vystřeluje 1 střelu.



Tato příšera má 2 životy a vystřeluje 2 střely.



Tato příšera má 3 životy a vystřeluje 3 střely.

Ve hře se vyskytuje UFO , jehož zabitím získáváme výhody (a jednu nevýhodu). Tyto výhody nazýváme Power-Ups (a Power-downs) a jsou to:



- Získáme štít, tj. pokud nás trefí střela nepřítele, nic se nám nestane



- Zrychlí se nám střílení, tj. můžeme střílet jakýsi roj střel.



- Místo jedné střely, střílíme 3 vedle sebe.



- Jediný Power-down ve hře. Obrátí se nám ovládání.

Do dalšího levelu se posuneme, pokud vystřelíme všechny příšery v hracím okně.

Konec hry nastane, pokud nás zasáhne střela příšery. V takovém případě se objeví okno s „Game Over“ a hra končí. Pokud chce hráč dále pokračovat ve hraní (samozřejmě od začátku), stačí stisknout ENTER.

# Programátorská dokumentace

## Zadání

Zadáním bylo vytvořit jednu z klasických her tak, aby byla něčím zajímavá, aby ji hráč mohl ovládat a příšery se pohybovaly, střílely a mizely po zabití.

Jednou z výzev bylo naučit se s novou knihovnou SFML (Simple and Fast Multimedia Library), která mi poskytla uživatelské rozhraní.

## Program

Program je tvořen hlavním souborem main.cpp a šesti soubory .cpp (Player, Enemy, EnemyManager, Ufo, Animation, DrawText) a dále devíti hlavičkovými soubory .h (Global, Player, Enemy, EnemyManager, Ufo, Animation, DrawText, Bullet, Powerup).

Soubory jsou takto rozděleny, aby byla zachována přehlednost a snadná úprava, případné dodělávky. Jména souborů jsou vypovídající, jen bych upřesnila pár detailů.

Rozdělila jsem Enemy na Enemy a Enemy Manager, abych zachovala přehlednost, neboť Enemy má nejvíce funkcí. EnemyManager má za úkol řídit střelbu, animace enemies, pohyb. Soubor Animation zodpovídá za jednotlivé malé animace, jako je například výbuch po zasažení střelou, změna barvy při získání Powerups, přeblikávání Powerupu, pohyb Ufa a pohybování enemies.

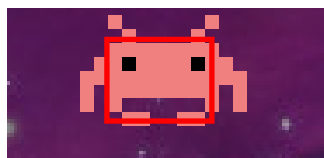
V souboru Global.h najdeme všechny konstanty.

Hlavní struktury jsou přehledně v hlavičkových souborech.

## Algoritmy

Srážka kulky a entity (vesmírná loď/enemy):

pokud se obdélník dané velikosti střetne s obdélníkem kulky, tedy když dojde k průniku dvou obdélníků.



Pohyb vesmírné lodi:

Lod' má nastavené počáteční souřadnice ve středu dole v okně. Pohybuje se tak, že pokud zmáčkne příslušnou klávesu, tak se k x-ové souřadnici přičtou, nebo odečtou 2 pixely. (Pokud chceme pohyb zrychlit, stačí v Global.h změnit konstantu player\_speed). Pokud se loď nachází na okraji okna (ten je nastaven na 16 pixelů od skutečného konce okna), tak se již tímto směrem nemůžeme dále pohybovat, tedy se zastavíme.

Okno: Pomocí SFML knihovny a knihovny chrono (práce s časem) je zařízení, aby se okno zavřelo, jen když mi rozhodneme, tj. program běží, dokud ho sami neukončíme.

Práce se Sprity a s Textures:

K vyobrazení obrázku do okna využíváme dva elementy, a to sprite a texture. Pokud chceme zobrazit daný obrázek, nejprve musíme vytvořit příslušný sprite a poté daný texture, tím se obrázek graficky promítne do herního okna. (viz main.cpp).

Levely:

Pro daný level je vždy vytvořen samotný .txt soubor, ve kterém je zapsáno schéma (nebo jakási mapka) rozložení, počet a druh enemies. 0 značí oranžového, 1 značí modrého, 2 značí fialkového. Poté, co se schéma načte, tak jednotlivé znaky (0/1/2) přetvoříme do příslušného druhu. Proto máme jako jeden z atributů třídy Enemy také typ.

## Animace

Ve hře narazíme na řadu drobných animací.

- Srážka kulky nepřítelů a lodi znamená, že se na 1 sekundu objeví červený výbuch. Provedení je takové, že z knihovny obrázků načteme obrázek výbuchu, což je několik zvětšujících se koleček vedle sebe, ty obarvíme na červenou a po obrázku přeskakujeme o velikost kolečka směrem doprava, čímž vytvoříme efekt výbuchu.
- Běžící Power-Up mění barvu, tj. pokud jsme zasáhli Ufo a to vypustilo modrý Powerup, bude přeblikávat světle a tmavě modře. Provedení je obdobné jako u předchozího. Máme k dispozici obrázek, na němž je vedle sebe tmavý a světlý powerup. Po tomto obrázku se posouváme o velikost jednoho powerup. To se děje neustále dokola, dokud powerup nechytíme, nebo pokud neopustí okno, tj. pokud jeho y-ová souřadnice není shodná s y-ovou souřadnicí velikostí okna.
- Pohybující se Ufo přeblikává. Opět máme vytvořen obrázek s několika variantami ufa a po tom se pohybujeme.
- Pohybující se enemies. Stejný princip jako výše.
- Pokud zasáhneme kulkou enemy, pak se daný enemy obarví na žlutou a zase zpět na svou původní barvu.
- Exploze Ufa

## Struktury

Class Player

Atributy

- bool dead – 1 mrtvý, 0 živý
- bool dead\_animation
- bool shield\_animation\_over – 0 pokud nemáme štít, 1 pokud máme
- char current\_power – podle daného powerupu, máme nějaký power, výchozí je 0
- char reload\_timer
- short power\_timer – jak dlouho budeme mít powerup
- short x – xová souřadnice
- short y – yová souřadnice

Metody

- bool get\_dead() - vrátí 1 pokud je mrtvý, jinak 0
- bool get\_dead\_animation()
- char get\_current\_power() - vrátí číslo daného powerupu ( 1 = shield, 2 = rychlejší střílení, 3 = střílení tří kulek vedle sebe, 4 = otočení ovládání)
- short get\_power\_timer()
- short get\_y() - vrátí yovou souřadnici

- void die() – vrátí 1 = smrt
- void draw() - vykreslí player
- void reset() - pokud je zasažen, všechny vlastnosti se vrátí do původního stavu
- void update() – zajišťuje změnu player při získání powerupu
- get\_hitbox() – vrací obdélníček player (tzv. hitbox), kde může být player zasažen

## Class EnemyManager

### Atributy:

- short Move pause – při spuštění hry, se na chvíli všechny enemies zastaví a teprve poté se začnou pohybovat
- bool reached\_player – 1 – pokud je zasažen hráč a hra končí, 0 pokud není zasažen

### Metody

- void draw – vykreslí daný typ enemy
- void reset – při změně levelu se předělá schéma enemies, načte se nový .txt soubor, rozestaví se do výchozí pozice
- void update – zajišťuje pohyb

## Class Enemy

### Atributy

- int direction – 0-Down, -1- left, 1- right
- char health – počet životů
- hit timer – pokud je zasažen, na tuto dobu se zobrazí žlutě
- char type – 0-oranžový, 1 – modrý, 2 - fialový
- short x – x-ová souřadnice
- short y – y-ová souřadnice

### Metody

- get\_health() – vrátí počet život
- get\_hit\_timer() – vrátí dobu pro změnu barvy na žluto po zasažení
- get\_type() – vrátí typ enemy
- move() – zajišťuje pohyb enemies, po okně
- shoot() – podle typu enemy vystřelí počet kulek
- update() – pokud je enemy zasažen, sníží jeho počet životů

## Class Ufo

### Atributy:

- bool dead
- bool dead\_animation\_over
- short explosion
- short x – xová souřadnice
- short y yová souřadnice
- short timer

### Metody

- bool check\_bullet\_collision() – zjistí zda bylo ufo zasaženo
- char check\_powerup\_collision() – zjistí zda se srazil player s powerupem
- void draw – vykreslí (viz animace)
- void reset – pokud se ufo dostane až na konec okna, resetuje se
- void update

- get\_hitbox

#### Class Bullet

##### Atributy

- bool dead
- float real\_x
- float real\_y
- float step\_x – posun o xovou souřadnici
- float step\_y – posun o yovou
- short x – výchozí x-ová souřadnice
- short y – výchozí y-ová

##### Metody

- update() - pohyb
- get\_hitbox()

#### Class Powerup

##### Atributy

- bool dead
- short x
- short y
- char type

##### Metody

- Poweup
- Get\_hitbox

#### [Celkové zhodnocení práce](#)

Práce na této hře mě bavila. Není to má první hra, neboť v prvním ročníku jsem si za svůj zápočtový program (Python) také zvolila hru.

Náročné pro mě bylo seznámit se s novou knihovnou SFML, ale je intuitivní, a proto jsem práci s ní zvládla rychle. Poté mi již byla jen k užitku a jsem ráda, že jsem si nevybrala jinou z možností (ještě se nabízela GLFW knihovna).