Uživatelská a programátorská dokumentace

# Počítačová hra
# Hladovec Herbert

Předmět Pokročilé programování v jazyce C# NPRG038,
garant: Mgr. Pavel Ježek, Ph.D.,
cvičení 22bNPRG038x05, učitel: RNDr. Jan Pacovský

Zápočtový program (hra), letní semestr 2023
Kristýna Harvanová, 2. ročník Bc. studia

# Obsah

# 1 Stručný popis programu

Program „Hladovec Herbert“ je počítačová hra inspirovaná arkádovou hrou ZX Spectrum z roku 1982 s názvem „Hungry Horace“. Cílem hry je dosáhnout co nejvyššího skóre, kterého může hráč v průběhu hry nabývat různými způsoby, přičemž musí čelit různým překážkám.

# 2 Uživatelská dokumentace

V této části dokumentu se nachází jak základní popis práce se hrou, tak podrobné vysvětlení pravidel a zákonitostí hry. Uživatel má možnost pravidla buď sám v průběhu hry objevovat, či si je nastudovat z uživatelské dokumentace.

## 2.1 Průvodce uživatelským rozhraním

1. Při spuštění hry se objeví úvodní obrazovka s názvem a stručným popisem cíle hry.
2. Zároveň vyběhne vyskakovací okénko „Nápověda“, které uživateli řekne, jakým způsobem lze obnovit poslední rozehranou hru (pokud byla uložena).
3. Rovněž se zde nachází ovládací prvky, které fungují následovně:
    a. Při kliknutí na tlačítko „KONEC“ se hra ukončí a automaticky dojde k zavření okna aplikace.
    b. Při kliknutí na tlačítko „HRA“ se objeví vyskakovací okénko s informacemi, jakým způsobem lze uložit rozehranou hru. Následovně se zobrazí první úroveň a hra se ihned spustí.
4. Uživatel má kdykoliv v případě potřeby možnost si hru pozastavit stiskem klávesy mezerníku.
5. Po konci hry se uživateli zobrazí informace o jejím průběhu a rovněž tlačítka „HRA“ a „KONEC“, která mají stejnou funkci jako na úvodní obrazovce (viz bod č. 2).

## 2.2 Podrobná pravidla hry

V této kapitole se zaměříme jak na podrobnou funkčnost jednotlivých postav, tak na zákonitosti a pravidla zisku bodů. Strategii k dosažení nejlepších výsledků si volí sám hráč.

### 2.2.1 Cíl hry

Cílem hry je získat co nejvyšší skóre, které lze nadále překonávat a vylepšovat, neboť je prakticky neomezené. Body do skóre přibývají v průběhu hry různými způsoby a hodnota skóre se nikdy nesnižuje. V průběhu hry se postupuje jednotlivými úrovněmi, v rámci kterých lze nasbírat určitý počet bodů. K postupu do další úrovně není potřeba získat veškeré body z aktuální úrovně. Hra končí v momentě, kdy je hráčem ovládaná postava Herberta v situaci, ve které je dostižen nepřáteli a už nemá možnost úniku.

### 2.2.2 Postavy a jejich vlastnosti

Veškeré postavy se pohybují v bludišti všemi směry. Mohou procházet přes všechna pole, až na ta, na kterých se nachází zeď. Zdi jsou vyobrazené tmavší barvou, než jsou ostatní pole, přes která se dá procházet. Pozor, v průběhu hry bude docházet ke zrychlování pohybu postav.

#### *Herbert*

Po spuštění hry se Herbert (tedy postava, kterou ovládá hráč) nachází u vstupu do bludiště dané úrovně (reprezentováno mapou, jako pohled shora). Vstup je vyobrazen dvěma černými šipkami směřujícími doprava a zpravidla se nachází na levé straně bludiště. K ovládání pohybu Herberta jsou používány klávesy šipek na klávesnici (stisk dané šipky pohne Herberta daným směrem). Cílem Herberta je sbírat body pro zvyšování skóre a vyhýbat se nepřátelům. Pokud se nepřátelé dostanou až k Herbertovi, hra končí.

#### *Nepřátelé*

Postavy nepřátel mají na rozdíl od Herberta výchozí pozici v cíli bludiště. Ten je označen rovněž dvěma černými šipkami směřujícími doprava, ovšem nachází se vždy více vpravo. Jakmile hra započne, nepřítel se začne pohybovat směrem k Herbertovi a snaží se ho dopadnout. Druhý nepřítel se objevuje o něco později (se stejným cílem), a to v momentě, kdy má Herbert na kontě v dané úrovni více než 40 získaných bodů. Opět po dosažení určitého skóre se objevuje třetí nepřítel.

V průběhu hry může nastat moment, kdy nepřátelé přestanou být hrozbou. V tomto momentě změní svůj vzhled a po styku s Herbertem mu na kontě přibude 100 bodů. Následně ovšem vstupují do hry noví nepřátelé, a tentokrát ne z místa cíle bludiště, ale z jeho vstupu (tj. z výchozí pozice Herberta).

### 2.2.3 Zisk bodů do celkového skóre

Herbert má možnost získávat body do skóre různými způsoby. Nejčastějším způsobem zisku bodů je návštěva políček, kde se nachází čtyřlístky. Za každý sebraný čtyřlístek dostává Herbert 10 bodů do celkového skóre. Níže se nachází popis dalších možností, kdy dochází k zisku bodů.

#### *Speciální odměna*

Pokud Herbert dosáhne v dané úrovni více než 400 bodů, jeden z jeho nepřátel je nucen zanechat v bludišti odměnu. Herbert má možnost si odměnu vzít tak, že se dostane k políčku, na kterém nepřítel zanechal odměnu ve formě dortu, a získává za něj 100 bodů.

#### *Dárek*

V každém bludišti se pro Herberta nachází jeden dárek. Pokud se k němu Herbertovi podaří dostat, získá za něj 100 bodů a zároveň dojde ke změně podstaty nepřátel. Ti přestanou být hrozbou a po střetu s každým z nich získá Herbert dalších 100 bodů.

# 3 Programátorská dokumentace

Tato část dokumentu podrobněji popisuje fungování programu a obsahuje informace o užitých algoritmech, způsobu čtení dat aj. Jsou zde vysvětleny základní myšlenky řešení daného problému. Podrobnější popis fungování jednotlivých částí programu je pro přehlednost uveden pomocí komentářů přímo ve zdrojovém kódu.

Počítačová hra „Hladovec Herbert“ je naprogramována v jazyce C# jako okénková aplikace Windows Forms běžící na platformě .NET jak v systému Windows, tak v systémech Linux a macOS. Kromě zdrojového kódu rozděleného do jednotlivých tříd používá textové soubory a obrázky ve formátech PNG.

## 3.1 Dekompozice

Program je rozdělen do několika tříd podle jejich logického významu a využití. Všechny třídy můžeme rozdělit do jednotlivých skupin, podle jejich uplatnění ve hře (viz obrázek 1). První skupinou jsou třídy herních objektů, což jsou postavy ve hře, jež se mohou pohybovat v závislosti na uživateli. Do druhé skupiny můžeme zařadit třídy formulářových objektů, které zajišťují uživatelské rozhraní. Poslední skupinou jsou třídy „pomocné“, které umožňují přehlednou práci s programem.

### 3.1.1 Vztahy mezi třídami

Na obrázku 2 jsou vyobrazeny základní vztahy mezi třídami. Rovněž jsou zde zmíněny nejdůležitější funkce.

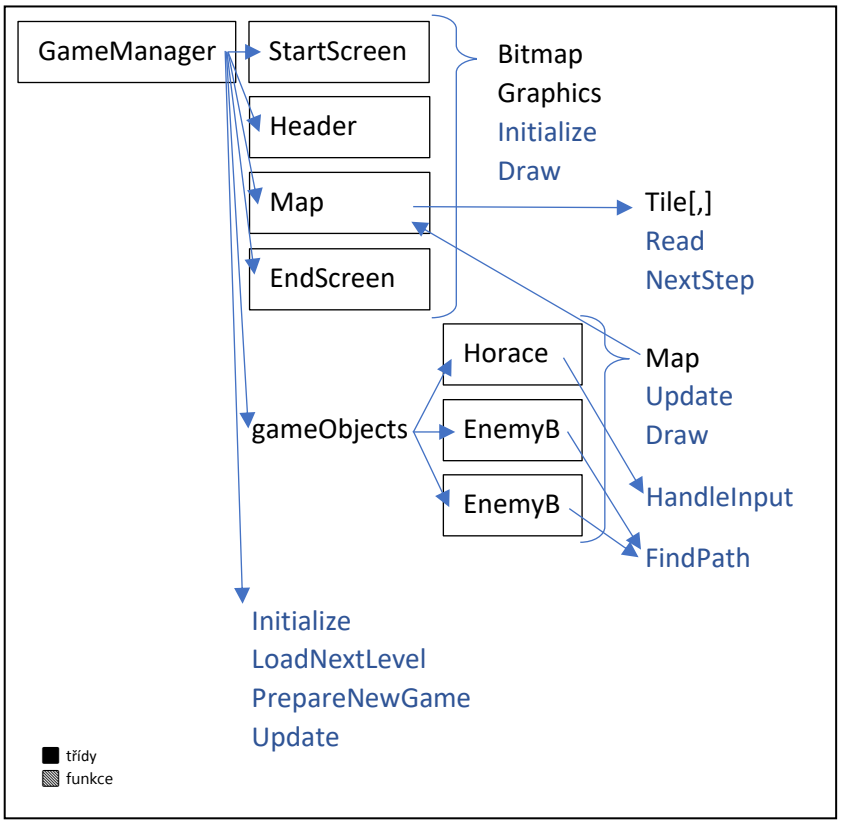
```
static class GameManager

abstract class GameObject
   abstract class Enemy
      class EnemyA
      class EnemyB
   class Horace

class FormObject
   class StartScreen
   class Header
   class Map
   class EndScreen

static class Input
class Tile
class VectorInt
class VectorFloat
class StateOfGame
classes in JsonConverters.cs
```

*Obrázek 1*
Přehled tříd




*Obrázek 2*
Vztahy mezi třídami

### 3.1.2 GameManager

Nejdůležitější třídou je statická třída „GameManager“. Ta má za úkol celou hru řídit, vidí tedy na všechny objekty a uchovává veškeré důležité konstanty.

Mezi základní funkce této třídy patří funkce „Initialize“, která při spuštění hry provede inicializaci všech nezbytných položek. Načte všechny nezbytné informace z textových souborů, připraví veškeré formulářové objekty, podle velikosti obrazovky daného zařízení přizpůsobí vykreslování hry a další.

Funkce „LoadNextLevel“ připraví prostředí a aktualizuje všechny důležité informace pro načtení další úrovně. Podobně funguje i funkce „PrepareNewGame“, která je spuštěna v momentě, kdy se uživatel po prohře rozhodne zahájit novou hru. Nemusí už dojít k prvotní inicializaci, stačí jen upravit část hodnot na výchozí. Nejčastěji volanou funkcí je funkce „Update“, která zobrazuje změny při vykreslování na obrazovku.

### 3.1.3 GameObject

Z abstraktní třídy „GameObject“ dědí všechny třídy, jež mají na starost pohyblivé objekty na mapě. Je to třída hlavního hrdiny Herberta ovládaného uživatelem (class Horace) a dále třída nepřátel (class Enemy), kteří se liší svými vlastnostmi, zejména cílem, k němuž směřují.

Každý herní objekt, kromě svého vzhledu, zná mapu, na které se pohybuje, a také místo svého výskytu na dané mapě. Dále si pamatuje rychlost svého pohybu. Aktuálně má hra 3 různé úrovně, tyto tři úrovně nazývejme jedním setem. Po úspěšném projití všech úrovní jednoho setu přichází nový set. Nový set je tvořen stejnými úrovněmi s tím rozdílem, že se objekty pohybují rychleji. Přitom je vždy dodrženo pravidlo, že je Herbert nepatrně rychlejší než jeho nepřátelé.

GameObject, kromě jiných, má vždy nezbytné funkce „Update“ a „Draw“, které zajišťují správnost vykreslování pohyblivých i nepohyblivých objektů na mapě vzhledem k nastalým událostem. Funkce „HandleInput“ třídy „Horace“ zpracovává vstup z klávesnice a patřičně upravuje hráčovu pozici na mapě. Spolu s dalšími funkcemi (testujícími např. zisk bodů vzhledem k aktuální pozici Herberta) tak zajištují správnost fungování nezbytné funkce „Update“.

Naproti tomu úlohou funkce „Update“ ve třídách „Enemy“ je pouze zjistit následující krok nepřítele tak, aby se pohyboval směrem ke svému cíli. Podrobněji je algoritmus vysvětlen v kapitole 3.2.1 Hledání nejkratší cesty. Je zde však nutno podotknout, že každý nepřítel hledá další krok své cesty ve skutečnosti k jinému cíli. Kdyby se totiž oba pohybovali na stejné místo, mohlo by se stát, že se jejich cesty protnou. Zároveň se také ve stejném okamžiku budou nacházet na stejném políčku a od této chvíle bude cesta, po které se pohybují, pro oba nepřátele stejná. V takovém případě by se nepřátelé pohybovali už vždy jen pospolu a nikdy by nedošlo k jejich oddělení, protože by vždy nacházeli společnou nejkratší cestu. Hráči by tak „ubyl“ jeden nepřítel, kterému se má vyhýbat a to nechceme.

Problém je vyřešen tak, že jeden nepřítel hledá nejkratší cestu na místo výskytu Herberta a druhý nepřítel hledá cestu k místu, které se nachází 4 políčka (nebo méně v případě překážky) před Herbertem. Pokud ovšem k tomuto políčku dorazí (hra nekončí, protože nebyl dopaden Herbert, ale jiné políčko), nepřítel se vydá hledat místo 4 políčka za Herbertem. Zdánlivě tedy dochází k efektu, že oba nepřátelé se snaží dopadnout přímo Herberta. Přitom je jen malá pravděpodobnost, že se jejich cesty spojí a nepřátelé splynou v jednoho a už se nikdy neoddělí.

### 3.1.4 FormObject

Objekt formuláře potřebuje znát svou velikost a další informace ke svému vykreslení. Každý objekt má svou vlastní bitmapu, na kterou je kreslený (opět pomocí funkce „Draw“), zejména pro možnost jednodušší práce s těmito formulářovými objekty. Jejich přesné rozměry jsou určeny při spuštění hry podle velikosti obrazovky zařízení uživatele.

Třídy „StartScreen“ a „EndScreen“ obsahují pouze statické informace pro uživatele a tlačítka k zadávání požadavků na hru („HRA“ či „KONEC“). Oproti tomu třídy „Header“ a „Map“ jsou vykreslovány v průběhu hry a informace v nich jsou měněny v reálném čase na základě progresu hry. V hlavičce se vyobrazuje aktuální skóre dosažené uživatelem. Mapa vykresluje bludiště a veškeré prvky v ní se nacházející.

### 3.1.5 StateOfGame a JsonConverters

Pro možnost uložení rozehrané hry a její obnovení i po vypnutí aplikace je využívána serializace do formátu JSON. Při uložení hry dojde k vytvoření instance třídy „StateOfGame“, jejíž properties schraňují všechny důležitá data k pozdějšímu obnovení hry. Z důvodu používání vlastních nadefinovaných tříd, není možná serializace a deserializace všech vytvářených objektů pomocí defaultního nastavení System.Text.Json. Proto jsou vytvořeny třídy „*Class*Converter“ jako potomci generické třídy „JsonConverter<T>“, které obsahují pomocný kód k serializaci a deserializaci složitějších vytvořených objektů. Instance těchto tříd jsou přidány jako „options“ do parametrů „JsonSerializer“ při převádění z a do formátu JSON.

### 3.1.6 Další třídy

Zbývající třídy, jako například „VectorInt“ nebo „Tile“, a výčtové typy, např. „EnemyState“, slouží k přehlednější práci s třídami, zmíněnými výše.

Třída „Input“ má za úkol zpracovávat data, která program dostává z textových souborů. Mezi její funkce patří například „ReadInt“ aj. Kromě čistého čtení dat má třída i speciální funkce pro zápis a čtení kódovaných dat. Tyto funkce překládají celočíselné hodnoty do zdánlivě náhodných znaků a naopak. Podrobněji popsáno v kapitole 3.2.2 Uchovávání informací o nejvyšším dosaženém skóre.

## 3.2 Užité algoritmy a jiná řešení problémů

Kapitola shrnuje základní informace použitých algoritmů. Zahrnuje i popis řešení dalších problémů.

### 3.2.1 Hledání nejkratší cesty

Algoritmus pro nalezení nejkratší cesty používají nepřátelé, jak je popsáno výše v kapitole 3.1.3 GameObject. V každém „Updatu“ hry se volá algoritmus znovu, protože Herbert v závislosti na uživateli mění svou pozici a nepřítel je nucen nalézt novou cestu, po které se má za svým cílem vydat. K tomuto účelu používáme algoritmus známý obecně jako A*.

Algoritmus A* je založen na Dijkstrově algoritmu hledání nejkratší cesty v grafu. Liší se tím, že používá heuristiku a najde tak vhodný výsledek v krátkém čase. Rovněž není potřeba, aby algoritmus zjišťoval délku nejkratších cest z výchozího místa (start) do všech ostatních. Stačí nám nalézt nejkratší cestu do jednoho konkrétního místa (cíl). K popisu algoritmu je důležité si uvědomit, že se můžeme pohybovat ve směrech nahoru, doprava, doleva a dolů a všechna tato sousední políčka (ta, na která můžeme vstoupit jedním krokem) mají od momentálního, na kterém se vyskytujeme, vzdálenost 1 (v jednotkách políček).

Pro každé navštívené políčko si budeme pamatovat jeho cenu, tedy počet políček, kterými jsme museli od startu projít, abychom se do něj dostali. Dále v průběhu pro každé nově navštívené políčko zjistíme jeho potenciálně nejkratší možnou cestu k cíli (předpokládáme, že na cestě nebudou žádné překážky), zde je uplatnění zmiňované heuristiky. Podrobnější popis algoritmu je uveden pomocí komentářů přímo ve zdrojovém kódu, kde je možno přesněji sledovat, co se ve danou chvíli děje.

### 3.2.2 Výpočty na separátních vláknech

S každým možným nepřítelem navíc který může přibýt narůstá objem výpočtů během jednoho časového tiku programu. Každá netriviální postava hry (ovládaná algoritmy složitými na výpočet) má tedy své vlastní vlákno, na kterém probíhají výpočty jejích časově i pamětně náročných algoritmů.

### 3.2.3 Uchovávání informací o nejvyšším dosaženém skóre

Abychom mohli uchovat nejvyšší dosažené skóre i po ukončení běhu aplikace, ukládáme (a čteme) ho ze souboru „HighestScore.txt“. Ovšem pro bezpečnost a omezení šancí uživatele si skóre uměle změnit je třeba ho nějakým způsobem šifrovat. Tento problém řeší funkce „TranslateFromInt“ a „TranslateToInt“ ve třídě „Input“.

Informace o skóre je pouze celočíselná hodnota, která díky zvolenému průběhu hry (postupné zrychlování pohyblivých herních objektů) pravděpodobně nikdy nebude nabývat extrémně velkých hodnot. Číslo zašifrujeme takovým způsobem, že ho zobrazíme jako zdánlivě náhodné znaky. Nejjednodušším způsobem je pracovat přímo s kódy znaků v ASCII tabulce.

Jelikož je informace o nejvyšším skóre uložena jako typ int (velikosti 32 bitů) a nemůže nabývat záporných hodnot (maximální hodnota půjde zapsat do 16 bitů), bude nám stačit využít prvních 128 znaků tabulky. Bude tak možné zašifrovat 128^4 - 1 = 268435455 různých čísel. Skóre však přibývá po násobcích 10, maximální možná hodnota k zašifrování bude tedy 268435450, což je pro hodnotu skóre naprosto postačující horní hranice.

Překlad informace z čísla na znaky probíhá takovým způsobem, že pokud je číslo menší než počet znaků v ASCII tabulce (128), místo samotného čísla zobrazíme znak, který je daným číslem reprezentován. Pokud je číslo vetší nebo rovno 128, vyobrazíme ho již dvěma znaky, kdy první znak (jeho hodnota v ASCII tabulce) zobrazuje počet násobků 128 a druhý znak zobrazuje zbývající hodnotu menší 128. Dále s vyššími čísly pracujeme podobně.

Překlad zpět ze znaků na číslo probíhá obdobně. Postupně načítáme znaky ze souboru a přiřazujeme jim jejich číselnou hodnotu (případně vynásobenou mocninou 128) podle umístění v ASCII tabulce. Jak si můžeme všimnout, ve skutečnosti jde jen o převod mezi dvěma číselnými soustavami – desítkovou a stodvacetiosmičkovou, kde k vyobrazení znaků stodvacetiosmičkové soustavy používáme znaky ASCII tabulky.

Pro běžného uživatele bez výraznějších znalostí informatiky je tedy prakticky nemožné rozluštit, co které znaky znamenají a jakým způsobem by si mohl skóre pozměnit.

### 3.2.4 Struktura mapy úrovní

Každá mapa je tvořena políčky (Tile[ , ]), na kterých se herní objekty pohybují. Některá políčka mohou být speciálního typu (enum TileType). Podle tohoto typu políčka se na něm může nacházet nějaká odměna, za kterou hráč získá body, nebo zeď, formující bludiště. Políčky, na nichž se nachází zeď, nemůže žádný herní objekt procházet. Zpravidla jsou políčka tvořící zeď v jedné vrstvě (tj. například jeden řádek) a políčka, jimiž se dá procházet, jsou vyobrazena ve vrstvách dvou (např. 2 řádky těsně nad sebou).

Informace o bludištích, tedy mapách jednotlivých úrovní, načítáme z textových souborů. V jednom takovémto souboru se nachází pouze informace o jedné určité úrovni. Tedy jen o tom, co se nachází na jednotlivých políčkách mapy. Při čtení souboru ignorujeme všechny pro nás nepodstatné znaky. Je tak možné si udržovat určitou strukturu díky pomocným znakům, které při načítání mapy nebudou brány v potaz.

Informace o celé hře, tj. všech úrovních, načítáme z textového souboru „MapsLevels.txt“. Zde je uveden vždy název textového souboru, ze kterého máme úroveň načíst, společně s šířkou a výškou této konkrétní mapy. Toto řešení tak představuje velmi jednoduché rozšíření pro další přidané úrovně.

### 3.2.5 Přizpůsobení vykreslování dle rozměrů zařízení

Jak pro pěkné zobrazení okna aplikace hry, tak pro uživatelsky příjemné herní prostředí udržujeme veškeré velikosti, rozměry a pozice formulářových objektů pouze relativní. Všechny tyto hodnoty uchováváme ne v jednotkách pixelů, ale v jednotkách políček. Prvotním cílem totiž bylo přizpůsobit velikost nejmenšího prvku mapy (jednoho políčka) tak, aby bylo vzhledem k danému zařízení přiměřeně velké. Nejmenší prvek mapy zde tedy přebral i roli metrické jednotky, která slouží nejen pro vykreslení samotného bludiště, ale poměrově i ostatních formulářových objektů.

Při načítaní formuláře zjistíme velikost obrazovky daného zařízení a z ní spočítáme, jakou velikost v pixelech chceme mít pro jedno políčko (konstanta „tileSize“ třídy „GameManager“). Z této hodnoty pak určíme velikost hlavičky i mapy, pozice tlačítek, textu aj. v bitmapě objektu. V neposlední řadě jsme schopni spočítat i přesný výskyt daných objektů na obrazovce tak, aby byly vycentrovány.