Karlova Univerzita

Matematicko-fyzikální fakulta

1. ročnik, Informatika

Započtový program

Programování 2

**«Pacman 2D»**

By Salohub Illia

Table of contents

Content

[Task description 4](#_Toc139640788)

[Project technologies 5](#_Toc139640789)

[Game Classes 6](#_Toc139640790)

[1. Game Manager Classes 6](#_Toc139640791)

[1.1. Game Manager Class 6](#_Toc139640792)

[1.2. Movement Class 7](#_Toc139640793)

[1.3. Animated Sprite Class 7](#_Toc139640794)

[2. Pacman classes 8](#_Toc139640795)

[2.1. Pacman 8](#_Toc139640796)

[2.2. Pellet 8](#_Toc139640797)

[2.3. Giant Pellet 8](#_Toc139640798)

[2.4. Portal 9](#_Toc139640799)

[3. Ghost Classes 9](#_Toc139640800)

[3.1. Default Ghost Class 9](#_Toc139640801)

[3.2. Class Ghost Behavior 10](#_Toc139640802)

[3.3. Class Node 10](#_Toc139640803)

[3.4. Class Ghost Scatter 11](#_Toc139640804)

[3.5. Class Ghost Chase 11](#_Toc139640805)

[3.6. Class Ghost Home 13](#_Toc139640806)

[3.7. Class Ghost Frightened 13](#_Toc139640807)

[3.8. Class Eyes 14](#_Toc139640808)

[4. Settings Classes 14](#_Toc139640809)

[4.1. Settings Men Class 14](#_Toc139640810)

[4.2. Algorithm setting 14](#_Toc139640811)

[4.3. Ghost speed setting 15](#_Toc139640812)

[4.4. Difficulty setting 16](#_Toc139640813)

[Comparison of the chosen algorithms 17](#_Toc139640814)

[Simple Algorithm 17](#_Toc139640815)

[Simple+ Algorithm 18](#_Toc139640816)

[BFS 18](#_Toc139640817)

[Conclusions 21](#_Toc139640818)

[Conclusions about the created game 23](#_Toc139640819)

# Task description

Pomocí herního enginu Unity s jazykem C# vytvořit 2D hru Pacman s implementací různých algoritmů pro Ghosts, hlavní požadavky jsou:

1. Vytvořit nastavení hry Pacman s mapou a všemi standardními objekty (pelety, obří pelety, duchové, pacman, portály, domov duchů) a herní smyčkou.

2. Vytvořit algoritmy pro duchy pro hledání Pacmana na mapě.

3. Vytvořte možnost měnit vlastnosti hry (rychlost duchů, algoritmy pro duchy atd.).

4. Vytvořit okno s pravidly hry

# Project technologies

Pro vytvoření Pacmana byly zvoleny další technologie:

- Unity 2022.3.2f1 + knihovna C# Unity Engine.

- C# 10.0 (.NET 6.0).

Unity je nejoblíbenější herní engine pro tvorbu jednoduchých i složitějších her pomocí programovacího jazyka C#.

Unity poskytuje jednoduchý rámcový systém pro tvorbu her na úrovni 3D her, her z pohledu třetí osoby, 2D a umožňuje vytvářet hry pro jakoukoli platformu, kterou chceme (Android, IOS, Linux, Windows, webové hry, PlayStation atd.).

# Game Classes

## 1. Game Manager Classes

Třídy správce hry obsahují logiku interakce mezi herními objekty, herní události, logiku pohybu a animace jednotlivých živých herních objektů.

### 1.1. Game Manager Class

Kód třídy Game Manager najdete zde.

Tato třída představuje události ve hře a logiku jejich spouštění.

Metoda New Game spustí novou hru, pokud byl pacman sněden nebo uživatel spustil aplikaci.

Изображение выглядит как снимок экрана, мультфильм, Красочность

Автоматически созданное описаниеMetoda PelletEaten volá metody pro získání pelet a pokud je pacman všechny snědl. Pokud pacman snědl ducha, pak třída volá metody pro návrat ducha domů, pokud duch snědl pacmana, pak třída volá metody pro ukončení herní relace a spuštění nové hry. (Viz video 1.1.1)

Video 1.1.1

### Movement Class

Kód třídy Pohyb najdete zde.Tato třída představuje změnu polohy pohyblivých objektů (Moving). Metoda používá metodu FixedUpdate pro aktualizaci pohybu objektů v reálném čase, také pomáhá provádět fyzikální výpočty, jako je detekce kolizí (detekce dotyků mezi objekty). Také metoda SetDirection pomáhá nastavit směr, pokud je dlaždice v daném směru dostupná, jinak ji nastavíme jako další směr, takže se automaticky nastaví, až bude dostupná, aby se objekty otáčely v daném směru, jen pokud je dostupný.

### Animated Sprite Class

Kód třídy Animated Sprite najdete zde.

Tato třída představuje animaci herního objektu (například: pacman otevírá a zavírá ústa, pacman umírá).

Používá seznam spritů a v metodě Advance code iteruje tento seznam a v každém pevně stanoveném časovém intervalu změní sprite tak, aby vypadal jako animace. (viz video 1.3.1)

Video 1.3.1

## 2. Pacman classes

Třídy Pacman obsahují logické chování Pacman pod kontrolou uživatele. Interakce mezi některými herními objekty (kromě duchů).

### 2.1. Pacman

Kód třídy Pacman najdete zde.

Tato třída obsahuje 3 metody: první metoda (Update) aktualizuje směr pohybu pacmana v závislosti na uživatelském vstupu z WASD nebo šipek (nahoru, dolů, doleva, doprava).

Druhá metoda ResetPacman se volá, když dojde k nějaké události (NewGame, NewRound), aby se resetovala pozice pacmana.

Poslední metoda DeathSequence slouží ke spuštění animace smrti, když pacman něco sní.

### 2.2. Pellet

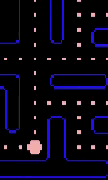
Kód třídy Pellet najdete zde.

Tato třída slouží k detekci dotyku mezi pacmanem a peletami, a když se dotýkají - skript volá metodu PelletEaten ze třídy GameManager a v této události se aktualizuje skóreИзображение выглядит как снимок экрана, мультфильм, Красочность

Автоматически созданное описание.

Video 2.2.1

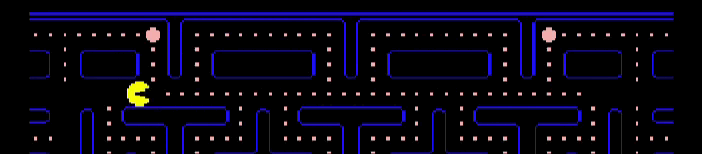
### 2.3. Giant Pellet

 Kód třídy Giant Pellet najdete zde.

Tato třída dědí třídu Pellet a přebírá metodu Eat, takže když ji Pacman sní, zavolá událost GiantPelletEaten ze třídy GameManager a všichni duchové se přepnou do režimu strachu, takže je Pacman může sníst (viz video 2.3.1).

Video 2.3.1

### 2.4. Portal

Kód třídy Portal najdete zde.

Video 2.4.1

Tato třída využívá 2 vstupy do portálů a teleportuje pacmana do jiného z nich, pokud mezi nimi a pacmanem existuje kolize. (Video 2.4.1)

## 3. Ghost Classes

Třídy duchů obsahují veškerou logiku umělé inteligence duchů. Tyto třídy popisují chování duchů během hry, jejich pohyby a algoritmy hledání pacmana.

### 3.1. Default Ghost Class

Kód třídy Ghost najdete zde.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание Tato třída slouží k nastavení objektu Ghost a kombinuje třídy chování v této entitě. Tato třída také obsahuje metodu reset ghost, která resetuje chování ducha při spuštění hry, a metodu pro detekci kolize mezi pacmanem a duchem a volání konkrétní události ze třídy Game Manager.

Image 3.1.1

### 3.2. Class Ghost Behavior

Kód třídy Ghost Behavior najdete zde.

Tato třída slouží k nastavení virtuálních metod Povolení/Zakázání chování duchů v čase, které budou přepsány ve všech třídách chování duchů.

### 3.3. Class Node

Изображение выглядит как снимок экрана, Красочность, схема

Автоматически созданное описание Kód třídy Node najdete zde.

Uzly jsou neviditelné objekty na mapě, které jsou umístěny na větvících se cestách, kde se aktualizuje logika některých duchů (cesta k pacmanovi v režimu honičky, cesta od pacmana v režimu strachu, cesta honičky v režimu rozptylu). (viz obrázek 3.3.1)

Image 3.3.1

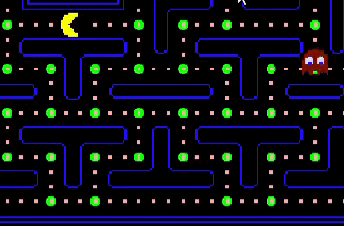
Изображение выглядит как Красочность, снимок экрана

Автоматически созданное описаниеUzly se kvůli optimalizaci nacházejí na větvících se cestách, takže algoritmy se aktualizují, když duchové dosáhnou větvících se cest.

Třída Uzel obsahuje seznam dostupných cest na větvících se cestách a algoritmy jej používají jako seznam dostupných pohybů pro duchy. (Video 3.3.2) Video 3.3.2

### 3.4. Class Ghost Scatter

Kód třídy Ghost Scatter najdete zde.

 Scatter Mode je specifický režim, kdy duchové na Pacmana neútočí. V tomto režimu se potulují kolem a to je vše, ale po určitém intervalu rozptylu se jejich režim změní na režim pronásledování a začnou pacmana pronásledovat.

V kódu si duchové v tomto režimu vybírají náhodnou cestu, když se dostanou do kolize s uzly, takže se diví a neútočí na pacmana.

### 3.5. Class Ghost Chase

Kód třídy Ghost Chase najdete zde.

Režim pronásledování je specifický režim, kdy duchové sledují pacmana pomocí specifického algoritmu hledání krátké cesty. V tomto režimu si duchové během kolizí s uzly vybírají nejlepší (podle názoru algoritmů) cestu, jak se dostat k Pacmanovi. Nyní existují 3 algoritmy, z nichž 2 jsou si velmi podobné.

Изображение выглядит как Красочность, снимок экрана, Графика, графический дизайн

Автоматически созданное описание První algoritmus ("Jednoduchý algoritmus") projde všechny dostupné směry aktuálního uzlu, vypočítá kolmou vzdálenost k pacmanovi z každého dostupného uzlu a vybere uzel s minimální vzdáleností k pacmanovi. (Viz obrázek 3.5.1)

Image 3.5.1

Изображение выглядит как снимок экрана, Красочность, Графика

Автоматически созданное описание Druhý algoritmus ("Simple+ Algorithm") funguje podobně jako jednoduchý algoritmus, ale počítá kolmou vzdálenost k uzlu před pacmanem o 2 buňky. Tento algoritmus se více podobá algoritmu z výchozí hry PacMan a je lepší než jednoduchý algoritmus, protože duchové hledají cestu k bodu před Pacmanem a v některých situacích mohou najít cestu k Pacmanovi rychleji, protože přemýšlejí o pozici, ve které bude Pacman, nikoliv ve které je. (Viz obrázek 3.5.2, šedý kruh - poloha cíle pro červeného ducha).

Image 3.5.2

Třetí algoritmus ("Breadth-first search(BFS)") funguje lépe než všechny předchozí algoritmy, protože všechny předchozí algoritmy počítají pouze jeden tah své cesty, obvykle to může být nejhorší cesta (viz obrázek 3.5.3).



Image 3.5.3(Simple algorithm)

BFS vypočítá úplnou cestu od ghosta k pacmanovi, takže najde nejkratší cestu k cíli a ghost se vydá správnou cestou.

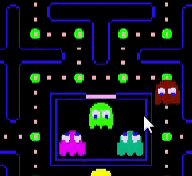


(see image 3.5.4)

Image 3.5.4 (BFS)

### 3.6. Class Ghost Home

Kód třídy Ghost Home najdete zde.

 Třída Ghost Home představuje chování ducha doma. Každý duch opouští domov v určitý čas (pro každého ducha je tento čas individuální), a když odchází - přechází do režimu Scatter Mode (viz video 3.6.1).

Také když pacman sežral ducha v režimu vyděšení - duch jde do domova a v určitém čase odchází.

Třída Ghost Home má pouze 2 specifické metody - metodu pro odraz od zdi a metodu pro opuštění domova.

Video 3.6.1

### 3.7. Class Ghost Frightened

Kód třídy Ghost Frightened najdete zde.

Изображение выглядит как снимок экрана, Красочность, прямоугольный, шаблон

Автоматически созданное описаниеRežim Frightened je specifický režim, kdy duchové nemohou jíst pacmana, ale pacman může jíst duchy. Duchové přejdou do režimu vyděšení na 7-10 sekund, když pacman sní obří peletu. Poté, co pacman sní vystrašené duchy - přejdou do režimu domů. (Video 3.7.1)

Během vystrašeného režimu se duchové na uzlech snaží před pacmanem utéct. Vybírají si dostupný uzel, který je od pacmana nejdále.

(Video 3.7.1)

Video 3.7.1

### 3.8. Class Eyes

Изображение выглядит как снимок экрана, Красочность, пиксель

Автоматически созданное описание Kód třídy Eyes najdete zde.

Tato třída slouží k nasměrování očí ducha v závislosti na směru jeho pohybu. (viz video 3.8.1)

Video 3.8.1

## 4. Settings Classes

Třídy nastavení jsou třídy s metodami, které otevírají menu hry a zastavují hru.

### Изображение выглядит как текст, снимок экрана, Шрифт, число Автоматически созданное описаниеSettings Men Class

Kód třídy Nastavení najdete zde.

Tato třída slouží k otevření nastavení a zobrazení potřebných nastavení v závislosti na kliknutí na tlačítko nastavení (Pravidla, Nastavení), Ukončení hry nebo Obnovení aktuální hry. (viz obrázek 4.1.1)

Image 4.1.1

### Algorithm setting

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описаниеKód na třídu nastavení algoritmů najdete zde.

Tato třída je určena k aktualizaci aktuálního algoritmu vyhledávání duchů v závislosti na uživatelské honbě. (Viz obrázek 4.2.1)

Image 4.2.1

### Ghost speed setting

Kód třídy Nastavení rychlosti najdete zde.

Tato třída byla vytvořena pro aktualizaci rychlosti Ghosts v závislosti na vstupu uživatele. (viz obrázek 4.3.1)

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описание

Image 4.3.1

### Difficulty setting

Kód třídy Obtížnost najdete zde.

Tato třída byla vytvořena pro aktualizaci doby trvání režimů duchů v závislosti na uživatelské obtížnosti. ( viz obrázek 4.4.1)

Изображение выглядит как текст, снимок экрана, Шрифт, Цвет электрик

Автоматически созданное описаниеNapříklad pro obtížnost Easy mají všichni duchové 10 sekundový režim Scatter, 20 sekundový režim Chase (30 pro červené duchy) a 10 sekundový režim Frightened.

Ale v obtížnosti Hard je režim Frightened kratší o 4 sekundy, Scatter o 6 sekund a režim Chase pro červeného ducha trvá 2 minuty, zatímco pro ostatní duchy je to 40 sekund.

Image 4.4.1

# Comparison of the chosen algorithms

## Simple Algorithm

Společná práce Simple algoritmusu :

Pro každý dostupný směr:

Vypočítáme novou polohu přičtením dostupného směru k aktuální poloze ducha.

Vypočítáme vzdálenost mezi novou polohou a cílovou polohou (např. polohou Pacmana).

Porovnáme vzdálenost s minimální vzdáleností.

Pokud je vypočtená vzdálenost menší než aktuální minimální vzdálenost:

Aktualizujeme minimální vzdálenost na vypočtenou vzdálenost.

Pak Aktualizujeme směr na aktuální dostupný směr.

Po iteraci všech dostupných směrů vrátíme vybraný směr s minimální vzdáleností.

Tento algoritmus je velmi jednoduchý, naznačuje, že duch si vybere směr, který ho přiblíží k cílové poloze (např. Pacman). Porovnáním vzdáleností mezi potenciálními pozicemi ducha a cílovou pozicí duch určí směr, který minimalizuje vzdálenost, a podle toho se přesune.

## Simple+ Algorithm

Běžná práce algoritmu Simple+ vypadá jako algoritmus Simple, ale:

Nyní nepočítáme vzdálenost mezi novou pozicí a pacmanem, ale počítáme vzdálenost mezi novou pozicí a bodem před cílovou pozicí, kde bodová pozice je cílová pozice plus dvojnásobek směrového vektoru cíle.

Cílem je simulovat ducha, který hledá bod před Pacmanem, aby našel rychlejší cestu než přímé pronásledování Pacmana.

Tento upravený algoritmus zavádí koncept hledání bodu před Pacmanem za účelem potenciálního nalezení rychlejší cesty. Uvažováním bodu, který je o dvě buňky před aktuální pozicí Pacmana, může být duch schopen obejít překážky nebo zablokované cesty (protože většina překážek je široká 2 buňky) a dosáhnout Pacmana efektivněji.

Tento algoritmus však také počítá cestu pouze k prvnímu uzlu a neví, co bude za dalším uzlem, takže může pouze doufat, že mapa bude snadno skousnutelná, ale někdy může dělat špatná rozhodnutí a účinnost této úpravy se může lišit v závislosti na konkrétním herním scénáři a návrhu úrovně.

## BFS

Práce s BFS je poměrně jednoduchá, ale poskytuje nejlepší algoritmus hledání cesty než Simple a Simple+. Současný systém BFS funguje jako:

Pro zobrazení herního prostředí je vytvořena 2D mřížka zvaná cesta. Každá buňka v mřížce představuje pozici na herní mapě a obsahuje informaci o tom, zda je buňka přístupná, nebo ne.

Mřížka silnic je inicializována na základě herního prostředí, přičemž přístupné pozice jsou označeny jako true a nepřístupné pozice jako false.

Poté inicializujeme frontu, do které se ukládají pozice, které byly navštíveny během procházení BFS.

Dokud fronta není prázdná, algoritmus pokračuje v prozkoumávání pozic.

Při každé iteraci je z fronty odstraněna pozice a seznam navštívených pozic vedoucích k této pozici. Pokud je vyřazená pozice cílovou pozicí, procházení se ukončí a seznam navštívených pozic se uloží jako výstup. V opačném případě algoritmus zkontroluje sousední pozice a zařadí je do fronty pro další zkoumání.

Pokud je sousední pozice dostupná a nebyla dosud navštívena, je spolu s aktualizovaným seznamem navštívených pozic zařazena do fronty. Dostupná pozice je rovněž označena jako navštívená aktualizací silniční sítě. Procházení BFS pokračuje, dokud není nalezena cílová poloha nebo dokud nejsou prozkoumány všechny dostupné polohy.

Pokud je při procházení BFS nalezena cesta, je jako další tah pro ducha vybrána první pozice ze seznamu navštívených pozic. Duch určí směr pohybu na základě své aktuální pozice a vybrané další pozice. Směr pohybu je vrácen jako hodnota směrového vektoru, Duch se pak pohybuje zvoleným směrem k další pozici. Proces se opakuje a algoritmus BFS se znovu provede pro nalezení dalšího pohybu.

Pokud se tak nestane(teoreticky to není možné), přejdeme k nejbližšímu uzlu a přegenerujeme cestu, abychom pacmana znovu našli.

BFS nám umožňuje:

- Zaručenou nejkratší cestu: BFS je známý pro nalezení nejkratší cesty mezi dvěma body v neváženém grafu. Pokud cesta existuje (teoreticky existuje vždy), BFS ji najde. To zaručuje, že duch vždy zvolí nejkratší cestu k dosažení cílové pozice.

- Optimální pohyb: Díky tomu, že BFS v každém kroku bere v úvahu všechny sousední uzly, zkoumá graf způsobem "do šířky". Tento přístup může vést k efektivním vzorcům pohybu v mapě bludiště.

- Vyhnutí se nesprávnému výběru uzlu: Algoritmus BFS zabraňuje výběru nesprávných uzlů, které by vedly k delším cestám. Tím, že BFS zkoumá uzly ve vzestupném pořadí podle jejich vzdálenosti od výchozí pozice, zajišťuje, že ghost nevybere neoptimální uzel, který by k dosažení cíle vyžadoval delší cestu.

## Conclusions

Jednoduché a jednoduché+ algoritmy implementují jednoduchou logiku s jediným výpočtem, což je pro hru velmi optimální, zatímco BFS má více výpočtů. BFS je však efektivnější než ostatní dva. Pro každý algoritmus můžeme sestavit tabulku + a -.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm  Property | Simple | Simple+ | BFS |
| Časová složitost generování dalšího směru pohybu | O(1) | O(1) | O(N\*M)  Kde n je počet řádků, m je počet sloupců na mapě. |
| Počet výpočtů | N \* 3 pro n dostupných směrů vypočítá sčítání, velikost a porovnání. | N \* 3 pro n dostupných směrů vypočítá sčítání, velikost a porovnání. | N x M (grid initialization) + C x A (BFS traversal calculations),  kde N je počet řádků, M je počet sloupců, C je počet iterací v cyklu procházení BFS a A je počet přístupných pozic. |
| Obrana před zaseknutou situací (když se pacman pohybuje a algoritmus nemůže vybrat cestu k pacmanovi, takže se otočí zpět). | Worse  (Příklad -  vzdálenost od prvního uzlu je menší, ale vzdálenost cesty je delší a algoritmus ho vybere, takže se musí vrátit) | Better  (Příklad -  Algoritmus zvolí kratší cestu, protože hledá bod, který je o 2 buňky napřed.) | Better  (Příklad -algoritmus nejkratší cesty bude vypadat takto) |
| Správnost krátké cílové cesty | Situační, ale horší než ostatní dva | Situační, ale lepší než Jednoduché | Lepší než všechny, protože hledání plné cesty. |
| Dlouhá cílová cesta  správnost | Worse  (vypočítat cestu k nejbližšímu uzlu, přičemž počet uzlů může být větší) | Worse  (vypočítat cestu k nejbližšímu uzlu, přičemž počet uzlů může být větší) | Better  Vypočítá úplnou cestu a aktualizuje ji v uzlech) |
| Overall | Situační/horší | Situační | Better |

Vidíme tedy, že BFS je mnohem lepší než ostatní dva algoritmy při výpočtu cesty, ale může být horší při optimalizaci, protože prochází všechny sousední buňky a jejich sousedy a nepoužívá žádnou specifickou metodu, jako je cena buňky, váha atd.

# Conclusions about the created game

V průběhu této práce byly provedeny následující úkoly:

1) Byly zkoumány metody tvorby 2D hry pomocí C# OOP v herním enginu Unity.

2) Byla vyvinuta logika herních událostí a chování duchů.

3) Byly vyvinuty tři vyhledávací algoritmy pro režim pronásledování duchů.

4) Vyvinuté algoritmy byly porovnány a byly vyvozeny závěry o efektivitě.