Karlova Univerzita

Matematicko-fyzikální fakulta

1. ročnik, Informatika

Započtový program

Programování 2

**«Pacman 2D»**

By Salohub Illia

Table of contents

Content

[Task description 4](#_Toc139579085)

[Project technologies 5](#_Toc139579086)

[Game Classes 6](#_Toc139579087)

[1. Game Manager Classes 6](#_Toc139579088)

[1.1. Game Manager Class 6](#_Toc139579089)

[1.2. Movement Class 6](#_Toc139579090)

[1.3. Animated Sprite Class 7](#_Toc139579091)

[2. Pacman classes 8](#_Toc139579092)

[2.1. Pacman 8](#_Toc139579093)

[2.2. Pellet 8](#_Toc139579094)

[2.3. Giant Pellet 8](#_Toc139579095)

[2.4. Portal 9](#_Toc139579096)

[3. Ghost Classes 9](#_Toc139579097)

[3.1. Default Ghost Class 9](#_Toc139579098)

[3.2. Class Ghost Behavior 10](#_Toc139579099)

[3.3. Class Node 10](#_Toc139579100)

[3.4. Class Ghost Scatter 11](#_Toc139579101)

[3.5. Class Ghost Chase 11](#_Toc139579102)

[3.6. Class Ghost Home 13](#_Toc139579103)

[3.7. Class Ghost Frightened 13](#_Toc139579104)

[3.8. Class Eyes 14](#_Toc139579105)

[4. Settings Classes 14](#_Toc139579106)

[4.1. Settings Men Class 14](#_Toc139579107)

[4.2. Algorithm setting 15](#_Toc139579108)

[4.3. Ghost speed setting 15](#_Toc139579109)

[4.4. Difficulty setting 16](#_Toc139579110)

[Comparison of the chosen algorithms 17](#_Toc139579111)

[Simple Algorithm 17](#_Toc139579112)

[Simple+ Algorithm 18](#_Toc139579113)

[BFS 18](#_Toc139579114)

[Conclusions 21](#_Toc139579115)

[Conclusions about the created game 23](#_Toc139579116)

# Task description

Using game engine Unity with C# to make a Pacman 2D game with implementation of different algorithms for Ghosts, the main requirements is:

1. Make a Pacman game setup with map and all standard objects (pellets, giant pellets, ghosts, pacman, portals, ghosts home) and game loop
2. Create an algorithms for ghosts for searching Pacman on the map
3. Make an ability to change game properties (Ghosts speed, ghosts algorithms, etc.)
4. Create window with game rules

# Project technologies

For creating Pacman, next technologies were chosen:

* Unity 2022.3.2f1 + C# Unity Engine library
* C# 10.0 (.NET 6.0).

Unity is the most popular game engine for creating simple and more complex games using C# programming language.

Unity provides easy frame system for creating games on level of 3D games, third person games, 2D’s, and allow creating games for any platform we want (Android, IOS, Linux, Windows, Web game, PlayStation, etc.).

# Game Classes

## 1. Game Manager Classes

Game manager classes contain the logic of interaction between game objects, game events, logic of moving and animation of each live game object.

### 1.1. Game Manager Class

Code to Game Manager Class you can find here.

This class represents events in game and logic to start them.

Изображение выглядит как снимок экрана, мультфильм, Красочность

Автоматически созданное описаниеMethod New Game start a new game if pacman were eaten or user start an application.

Method PelletEaten call methods to obtain a pellet and if pacman has eaten all of them. If pacman has eaten ghost, then class call methods to return ghost to the home, if ghost has eaten pacman, then class call methods to end game session and start new game. (See video 1.1.1)

Video 1.1.1

### Movement Class

Code to the Movement Class you can find here.

This class represents changing position of movable objects (Moving). Method uses FixedUpdate method to updates movements of objects in real-time, also it helps doing physics calculations, such as collision detections(Detection touches between objects). Also method SetDirection helps set the direction if the tile in that direction is available, otherwise we set it as the next direction so it'll automatically be set when it does become available, to rotate objects in the direction only if it is available.

### Animated Sprite Class

Code to the Animated Sprite Class you can find here.

This class represents animation of game object (For example: pacman opens and closes his mouth, pacman death).

It uses List of sprites and in the method Advance code iterate through this list and every fixed time interval it changes sprite to make it looks like an animation. (see video 1.3.1)

Video 1.3.1

## 2. Pacman classes

Pacman classes contain the logic pacman behavior under user controlling. Interactions between some of game objects (Excluding Ghosts).

### 2.1. Pacman

Code to the Pacman Class you can find here.

This class contain 3 methods: first method (Update) update pacman movement direction, depending on user input from WASD or Arrows (Up, Down, Left, Right).

Second method ResetPacman is being called when some event happens (NewGame, NewRound) to reset pacman position.

Last Method DeathSequence is used to start animation of death when pacman has eaten.

### 2.2. Pellet

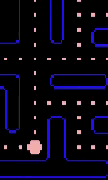
Code to the Pellet Class you can find here.

Изображение выглядит как снимок экрана, мультфильм, Красочность

Автоматически созданное описание This class is used to detect touching between pacman and pellets, and when the are touching – script call method PelletEaten from GameManager class, and in that event score are being updated.

Video 2.2.1

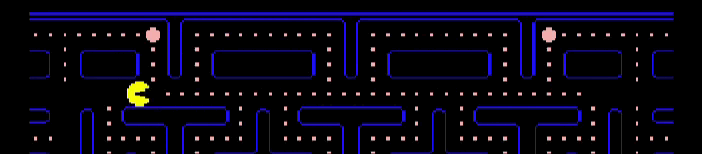
### 2.3. Giant Pellet

Code to the Giant Pellet Class you can find here.

This class inherits Pellet class and override method Eat, so when it’s being eaten by pacman, it calls event GiantPelletEaten from GameManager class and all ghosts go into frightened mode, so Pacman can eat them.(see video 2.3.1)

Video 2.3.1

### 2.4. Portal

Code to the Portal Class you can find here.

Video 2.4.1

This class takes 2 portal entrances to teleport pacman to another one of them, when collisions between them and pacman exists. (Video 2.4.1)

## 3. Ghost Classes

Ghost Classes contains all logic of AI of ghost. These classes describe behavior of ghosts during the game, their movements and pacman search algorithms.

### 3.1. Default Ghost Class

Code to the Ghost Class you can find here.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание This class is used to set up an object of Ghost and combines behavior classes in that entity. Also, this class contain reset ghost method, to reset ghost behavior when game starts, and method to detect collision between pacman and ghost and call specific event from Game Manager class.

Image 3.1.1

### 3.2. Class Ghost Behavior

Code to the Ghost Behavior Class you can find here.

This class is used to set virtual methods of Enabling/Disabling ghost behavior on time, that will be overridden in all ghost behavior classes.

### 3.3. Class Node

Изображение выглядит как снимок экрана, Красочность, схема

Автоматически созданное описание Code to the Node Class you can find here.

Nodes it’s invisible objects on the map, that are placed on branching paths, where some of ghosts logic are updating (Path to the pacman in chase mode, path from the pacman in frightened mode, way chase in scatter mode). (see Image 3.3.1)

Image 3.3.1

Изображение выглядит как Красочность, снимок экрана

Автоматически созданное описаниеNodes are located on branching paths because of optimization, so algorithms are being updated when ghosts reached the branching paths.

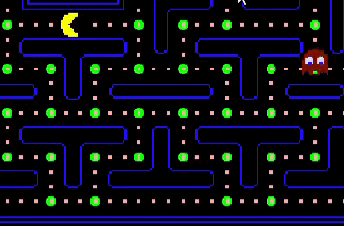
Class Node contains list of available paths on branching ways, and

algorithms use it as list of available movements for ghost. (Video 3.3.2)

Video 3.3.2

### 3.4. Class Ghost Scatter

Code to the Ghost Scatter Class you can find here.

Scatter Mode it is specific mode, when Ghosts don’t attack pacman. In that mode they are wondering around and that’s all, but after specific scatter interval their mode will be changed to chase mode and they will start following pacman.

In code, during this mode ghosts choose random way when they are have collision with Nodes, so it is why they wonder around and don’t attack pacman.

### 3.5. Class Ghost Chase

Code to the Ghost Chase Class you can find here.

Chase mode is a specific mode when ghosts follow pacman using the specific algorithm of short path finding. In that mode ghosts during the collisions with Nodes are choosing best (on opinion of algorithms) way to reach Pacman. Now there are 3 algorithms, where 2 are very similar.

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

Автоматически созданное описаниеFirst algorithm (“Simple Algorithm”) make iterate through all available directions of the current Node, calculate perpendicular distance to pacman from every available node and choose node with minimal distance to pacman. (See Image 3.5.1)

Image 3.5.1

Изображение выглядит как снимок экрана, Красочность, Графика

Автоматически созданное описание Second algorithm (“Simple+ Algorithm”) works similar to Simple algorithm, but calculate perpendicular distance to node in front of pacman by 2 cells. This algorithm is more like algorithm from default PacMan game, and it is better then Simple Algorithm because Ghosts are searching path to the point in front of Pacman and in some situations they can find path to pacman faster, because they think about the position in which pacman will be, not in which it is. (See Image 3.5.2, gray circle – position of target for red ghost)

Image 3.5.2



Third algorithm (“Breadth-first search(BFS)”) works better than all algorithms before, because all algorithms before calculate only one move of their path, usually it can be worst way (see image 3.5.3).

Image 3.5.3(Simple algorithm)

BFS calculate full path from ghost to pacman, so it finds the shortest path to target and ghost will go to the correct way.

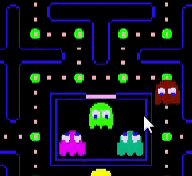


(see image 3.5.4)

Image 3.5.4 (BFS)

### 3.6. Class Ghost Home

Code to the Ghost Home Class you can find here.

 Ghost Home class represents Ghost behavior at home. Each ghost leave home at specific time (for each ghost this time is individual), and when they are leaving – they are went to Scatter Mode.(see video 3.6.1)

Also when pacman has eaten ghost in frightened mode – ghost go to the home, and in specific time he leaves.

Ghost Home class has only 2 specific methods – method to bounce from wall and method to leave home.

Video 3.6.1

### 3.7. Class Ghost Frightened

Code to the Ghost Frightened Class you can find here.

Изображение выглядит как снимок экрана, Красочность, прямоугольный, шаблон

Автоматически созданное описаниеFrightened Mode is the specific mode when ghosts can’t eat pacman, but pacman can eat ghosts. Ghosts go into frightened mode for 7-10 seconds when pacman has eaten giant pellet. After pacman has eaten frightened ghosts – they go to home. (Video 3.7.1)

During frightened mode, ghosts on Nodes try to run away from pacman. They choose available node that is farthest from the pacman.

Video 3.7.1

### 3.8. Class Eyes

Изображение выглядит как снимок экрана, Красочность, пиксель

Автоматически созданное описание Code to the Eyes Class you can find here.

This class is used to direct eyes of ghost depending on direction of ghost movement. (see video 3.8.1)

Video 3.8.1

### 

## 4. Settings Classes

Settings Class is class with methods that opens menu of the game and stop the game.

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

Code to the Settings Class you can find here.

This class is made for opening settings and display needed settings depending on the clicked settings button (Rules, Settings), Quit the game, or Resume current game. (see image 4.1.1)

### 

Image 4.1.1

### Algorithm setting

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

Автоматически созданное описаниеCode to the Algorithm setting class you can find here.

This class is made for updating current ghost search algorithm, depending on user chase. (See image 4.2.1)

Image 4.2.1

### Ghost speed setting

Code to the Speed Setting class you can find here.

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

Автоматически созданное описаниеThis class were made for updating Ghosts speed depending on user input. (see Image 4.3.1)

Image 4.3.1

### Difficulty setting

Code to the Difficulty class you can find here.

This class were made for updating ghosts modes duration depending on the user difficulty chase. ( see Image 4.4.1)

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

Автоматически созданное описаниеFor example for Easy difficulty all ghost has 10 second Scatter mode, 20 seconds Chase mode (30 for red ghost) and 10 seconds Frightened Mode.

But in Hard Difficulty Frightened Mode is shorter by 4 seconds, Scatter by 6 seconds, and Chase Mod for red ghost is 2 minutes, while for other ghosts it’s 40 seconds.

Image 4.4.1

# Comparison of the chosen algorithms

## Simple Algorithm

Common work of Simple algorithm :

For each available direction:

We Calculate a new position by adding the available direction to the current position of the ghost.

We Calculate the distance between the new position and the target position (e.g., Pacman's position).

We Compare the distance to the minimum distance.

If the calculated distance is less than the current minimum distance:

We Update the minimum distance to the calculated distance.

Then We Update the direction to the current available direction.

After iterating through all available directions, we return the chosen direction with the minimum distance.

This algorithm is very simple, it suggests that the ghost will choose the direction that brings it closer to the target position (e.g., Pacman). By comparing distances between the ghost's potential positions and the target position, the ghost will determine the direction that minimizes the distance and move accordingly.

## Simple+ Algorithm

Common work of Simple+ algorithm looks like Simple algorithm, but:

Now we don’t calculate distance between new position and pacman, but we calculate distance between new position and a point ahead of the target position, where point position is the target position plus twice the target's direction vector.

This aims to simulate the ghost looking for a point ahead of Pacman to find a faster path than directly pursuing Pacman.

This modified algorithm introduces the concept of looking for a point ahead of Pacman to potentially find a faster path. By considering a point two cells forward from Pacman's current position, the ghost may be able to navigate around obstacles or blocked paths(because most of the obstacles are 2 cells width) and reach Pacman more efficiently.

However, this algorithm also calculate a path only to the first Node, and it doesn’t know what will be after next Node, so it can only hope that map will be easy to bit, but can do bad decisions sometimes and effectiveness of this modification may vary depending on the specific game scenario and level design.

## BFS

Work of BFS pretty simple, but it give us best algorithm of pathfinding then Simple and Simple+. Current BFS work as:

A 2D grid called road is created to represent the game environment. Each cell in the grid represents a position in the game map and contains information about whether the cell is accessible or not.

The road grid is initialized based on the game environment, where accessible positions are marked as true and inaccessible positions are marked as false.

Then we initialize a queue, to store the positions that were visited during the BFS traversal.

While the queue is not empty, the algorithm continues to explore positions.

At each iteration, a position is dequeued from the front of the queue, along with the list of visited positions leading to that position. If the dequeued position is the target position, the traversal terminates, and the list of visited positions is saved as output. Otherwise, the algorithm checks neighboring positions to enqueue them for further exploration.

If a neighboring position is accessible and has not been visited before, it is enqueued along with an updated list of visited positions. The accessible position is also marked as visited by updating the road grid. The BFS traversal continues until either the target position is found or all accessible positions have been explored.

If the BFS traversal found a path the first position from the list of visited positions is selected as the next move for the ghost. The ghost determines the movement direction based on its current position and the selected next position. The movement direction is returned as a direction vector value, The ghost then moves in the chosen direction towards the next position. The process repeats, and the BFS algorithm is executed again to find the next move.

If not(theoretically this is impossible), then we go to the closest Node to regenerate path to find pacman again.

BFS allow us:

* Guaranteed shortest path: BFS is known for finding the shortest path between two points in an unweighted graph. If a path exists(theoretically it always exists), BFS will find it. This ensures that the ghost will always choose the shortest path to reach the target position.
* Optimal movement: By considering all neighboring nodes at each step, BFS explores the graph in a breadth-first manner. This approach can lead to efficient movement patterns in maze map.
* Avoidance of incorrect node selection: The BFS algorithm avoids choosing incorrect nodes that would lead to longer paths. By exploring nodes in increasing order of their distance from the starting position, BFS ensures that the ghost won't select a suboptimal node that would require a longer path to reach the target.

## Conclusions

Simple and simple+ algorithms implement easy logic with only one calculation, it is very optimizing for game, while BFS has more calculations. But BFS is more efficient than other two. We can make table of + and – of every algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm  Property | Simple | Simple+ | BFS |
| Time complexity to generate a next direction of move | O(1) | O(1) | O(N\*M)  Where n is number of rows, m is number of column on a map |
| Number of calculations | N \* 3 for n number of available directions it calculates addition, magnitude and comparison | N \* 3 for n number of available directions it calculates addition, magnitude and comparison | N x M (grid initialization) + C x A (BFS traversal calculations),  where N is the number of rows, M is the number of columns, C is the number of iterations in the BFS traversal loop, and A is the number of accessible positions. |
| Defense from stuck situation (when pacman is moving and algorithm can’t choose a path to pacman, so it turns back) | Worse  (Example –  distance from first node are less, but path distance are longer, and algorithm will choose it, so it has to turn back) | Better  (Example –  Algorithm will choose shorter path because it is searching for point that is ahead by 2 cells) | Better  (Example – algorithm shortest path will look like this) |
| Short destination path correctness | Situational, but worse then other two | Situational, but better then Simple | Better then all, because search for full path. |
| Long destination path  correctness | Worse  (calculate way o the nearest node, while number of nodes can be more) | Worse  (calculate way o the nearest node, while number of nodes can be more) | Better  (Calculate full path and update it on nodes) |
| Overall | Situational/Worse | Situational | Better |

So, we can see, that BFS is much better than other two algorithms in path calculating, but can be worse in optimization, because it goes to every neighbor cells and to their neighbors and don’t use any specific method like cell cost, weight, etc.

# Conclusions about the created game

In the course of this work, the following tasks was performed:

1)Were explored methods of creating 2D game using C# OOP in Unity game engine.

2)Game events logic and ghosts behaviors were developed.

3) Three search algorithms were developed for ghosts chase mode.

4) Developed algorithms were compared and conclusions about effectivity were reached.