C++ 2022A - MTA - Exercises

Requirements and Guidelines

The exercises in the course would require you to implement a pacman game as a console application.

Note: the exercise should be implemented in Visual Studio 2019 or later (e.g. VS 2021), with standard C++ libraries and run on Windows with Console screen of standard size (80*25), using gotoxy for printing at a specific location on screen (X, Y), using _kbhit and _getch for getting input from the user without blocking, and Sleep for controlling the pace of the game. Submission is in MAMA, as a single zip file containing only the code and the vcproj and sln files + readme.txt with IDs -- but without the DEBUG folder and without any compiled artifact.

Exercise 1

In this exercise you will implement the basic pacman game for a human player.
Read the pacman rules in the wiki link above, but you should support only the required features listed below.
You decide how to use the size of the screen for:

- Presenting the board.
- Presenting number of points and “remaining lives” information.

When the game starts the pacman is positioned at his start position (you should decide where it is) without any movement. Once the user selects move direction (using the keys, as listed below) the pacman will continue to move in this direction even if the user doesn’t press any key, as long as the pacman doesn’t hit a wall and the “STAY” key was not pressed.
Keys:

LEFT	a or A
RIGHT	d or D
UP	w or W
Down	x or X
STAY	s or S

Menu

The game shall have the following entry menu:
(1) Start a new game
(8) Present instructions and keys
(9) EXIT

Pausing a game

Pressing the ESC key during a game pauses the game. It *can be good* to present a message on screen saying: “Game paused, press ESC again to continue”. But it is not mandatory to present such a message.
When the game is at pause state, pressing ESC would continue the game, with the pacman continuing his movement exactly as it was before pausing, as if the game hadn’t paused.

The screen is consisted of:

- walls (you shall use any reasonable char to draw them)
- pacman (use @ or any other reasonable char)
- 2 ghosts (use $ or any other reasonable char)
- “breadcrumbs” on all unvisited positions (you shall use any reasonable char to draw them)

The pacman travels on screen and “eats the breadcrumbs”. Each eaten breadcrumb equals a point to be earned. Once all breadcrumbs on screen are eaten the game ends with a happy message of your choice. Pressing *any key* should get back to the main menu.

In case a ghost eats the pacman, the player loses one “life”. If all “lives” are gone, there should be a message announcing “Game Over” then after pressing *any key* the main menu shall be presented.

The ghosts hover above the breadcrumbs without eating them. You can decide how they go (no need to make them “smart”). Their pace is half the pace of the pacman (for any two steps of the pacman they make 1 step). You may decide if a ghost can go over the other ghost if they meet (i.e. both may share the same position) or not.

The pacman can cross the screen in an invisible tunnel from the rightmost position to the leftmost position of the same row and vice versa, and from the topmost position to the downmost position of the same column and vice versa, if there are no walls at both sides. Ghosts cannot cross in those invisible tunnels!

Note that there is one screen in Ex1, which looks the same for all games.

Pacman has 3 lives. After losing a life all creatures start from the exact initial position, but without the breadcrumbs that were eaten.

Bonus points: colors. Other nice features.
Note: there will not be any bonus for music or any feature that requires additional binary files to be part of your submission! Adding large required binary files to your submissions may subtract points!

Notes on Ex1

1. If you decided to add colors (as a bonus feature) please add an option in the menu to run your game with or without colors (the default can be to use colors, but the menu shall allow a switch between Colors / No Colors) - to allow proper check of your exercise in case your color selection would not be convenient for our eyes. The game MUST work properly in the No Colors selection.
2. Please indicate inside your readme.txt file the bonus additions that you implemented.

Exercise 2

In this exercise you will implement the following additions to your game:

A Fruit

A fruit is a single char “creature”, the char shall be a digit between 5 to 9 selected randomly, that appears at random points in time on screen and travels around randomly at a slow pace. If it meets a ghost it disappears. If the pacman eats the fruit it gets the points as the value of the digit (5 to 9).
The fruit disappears after some time.
There can be only 1 (or zero) Fruits on screen at a certain time.
Fruits cannot cross in the invisible tunnels!

Smarter Ghosts

Game shall have 3 possible levels (either in the main menu or after starting a game):
(a) BEST (b) GOOD and (c) NOVICE
BEST - Ghost try to chase the pacman
GOOD - Ghost try to chase the pacman, but occasionally (randomly, once in ~20 moves) they just change to a random direction and stay with it for 5 moves before being smart again
NOVICE - Ghost just move on screen with a direction selected randomly every 20 moves

Loading Screens from files

The game would look for files in the working directory, with the names *pacman_*.screen* these files would be loaded in lexicographical order (i.e. *pacman_a.screen* before *pacman_b.screen* or *pacman_01.screen* before *pacman_02.screen* etc.).
If there are no files, a proper message would be presented when trying to start a new game.
The menu should have a new option to allow running a specific screen, by name.
The screen file should be a text file representing the screen, with:
@ - for the position of the pacman (1 and only)
$ - for the initial position of the ghosts (allow any number between 0 to 4)
# - for walls
& - for the position where the points and remaining lives information shall be presented, it is the responsibility of the screen designer (not of your program) to make sure that the & is placed in a position not accessible by the creatures. You may assume that the screen designer follows this instruction.
Note that the above chars are mandatory, even if you use other chars for drawing the board on screen. Note also that the file doesn’t contain breadcrumbs and you should add them yourself when drawing the screen.

Pacman still has 3 lives, for all screens (moving to next screen doesn’t gain lives).

Note

You should change your original code, to use new materials that we learned - where appropriate.

Exercise 3

In this exercise you will implement an option to run a game from files and to record a game into files, mainly for testing. This would work as following:

- There would be a text file with a list of steps for all creatures, per screen in a structure of your choice. The structure shall be concise, i.e. shall include only change of movement directions and appearance / disappearance of a new Fruit. Name of file shall be in the format corresponding the screen name: *pacman_*.steps* (e.g. *pacman_01.steps*)
- There would be a file with the expected result for the screens, in the format corresponding the screen name: *pacman_*.result* (e.g. *pacman_01.result*), the file shall include the following information:
    - points of time for this screen where pacman died (if any)
    - point of time for this screen where pacman finished the screen

You have the freedom to decide on the exact format of the files but without changing the above info or adding unnecessary additional info.

You should provide at least 3 examples, for three different screens, keeping the lives of the pacman till the 3rd screen. You should also add a dedicated readme file called **file_format.txt** explaining your files format, to allow creation of new files or update of existing files.

The game shall be able to load and save your files!
All files shall be retrieved / saved from /to the current working directory.

Running the option for loading or saving game files would be done from the command line with the following parameters:

```
pacman.exe -load|-save [-silent]
```

The *-load/-save* option is for deciding whether the run is for saving files while playing (based on user input) or for loading files and playing the game from the files (ignoring any user input).
In the -load mode there is NO menu, just run the loaded game as is, **and finish!** Also you should ignore any user input, including ESC - there is no support for ESC in load mode!
In the -save mode there is a menu and the game is the same as in Ex2, except that files are saved. Note that each new game overrides the files of the previous game, i.e. we always keep the last game saved (you can always take the files and copy them to another folder annually if you wish).

The *-silent* option is relevant only for load, if it is provided in save mode you should ignore it, if it is provided in load mode, then the game shall run without printing to screen and just testing that the actual result corresponds to the expected result, with a proper output to screen (test failed / passed). In silent mode you should better avoid any unnecessary sleep! - the game should run as fast as possible. Without the -silent, loaded game being presented to screen, you may use smaller sleep values than in a regular game.

Note:
You should still support running your game in “simple” mode, without any command line parameters, as in Ex2: `pacman.exe`
In which case it will **not save or load files** and would behave as in Ex2.

Exercise 4

Exercise 4 is a separate exercise built as a rehearsal for the exam.
It would be published separately.