C++ 2022B - MTA - Exercises

Requirements and Guidelines

The exercises in the course would require you to implement a game inspired by the old and famous “Thunderbirds” game, as a console application.

Here are a few links that can get you acquainted with the game:
https://www.youtube.com/watch?v=bFJLZWDOA-M&t=21m12s (you can see the entire video if you have time, or just this part).
https://worldofspectrum.org/archive/software/games/thunderbirds-firebird-software-ltd
https://www.gamesdatabase.org/game/sinclair-zx-spectrum/thunderbirds
Original instructions:
https://www.gamesdatabase.org/Media/SYSTEM/Sinclair_ZX_Spectrum/Manual/formated/Thunderbirds_-_1989_-_Grandslam_Entertainment.htm

Note 1: the exercise is *inspired* by the “Thunderbirds” game, but is not exactly the same, so you should follow the requirements below.

Note 2: 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.

Thunderbirds
Two “ships” are trapped inside an ancient Egyptian tomb. A big one (size = 2*2) and a small one (size = 2*1). The big ship would be denoted on screen with # chars, or any other representation that you find suitable, the small ship would be denoted on screen with @ chars, or any other representation that you find suitable.
Note that in your game, the small ship cannot rotate, which means it is always in horizontal mode. The big ship also cannot rotate, but even if it could it has a rectangle shape so it doesn’t matter anyhow.
Both ships cannot move through walls. Walls would be denoted on screen with a special character of your choice.
Either than the walls, there are on screen “blocks”, that serve as obstacles for the ships. The blocks would be denoted on screen with a special character of your choice.
At any point of time you can move either the big or the small ship. The big ship can move or carry blocks of total size 6. The small ship can move or carry blocks of total size 2.
At each “screen” the ships start at a certain “entry” position (set by the screen designer, that is *you*, it would be a different entry position per each ship, as the ships cannot reside on the same location at the same time). The player shall finish the screen in a given “time” (set by the screen designer, that is *you*, time can be measured by arbitrary “ticks”, not necessarily seconds) by reaching the “exit” point on screen (set by the screen designer, that is *you*) with both ships. The screen should show the amount of time left to finish. If the ships do not finish on time, the player loses. The player has 3 lives, so if there are lives left, the screen would restart after losing. When restarting the screen the time limit is reset to its original value and also all blocks return to their original location.
When a block “falls” (after being pushed by a ship and reaching a “no floor” position) it would fall down at a pace that the game designer (*you*) decides. In case a block falls on a ship, if the block is in the size limits which the ship can hold, it will carry it, otherwise if it is too big, the ship “dies” and the player loses. Again, if there are lives left the screen would restart.
In Ex1 the game would have only one screen. In Ex2 it may have any number of screens.

Exercise 1

In this exercise you will implement the basic “Thunderbirds” game for a human player with a single screen.

You decide how to use the size of the screen for:

- Presenting the board.
- Presenting “remaining lives”, “Time left for this screen” and “active ship” information.

When the game starts the ships are positioned at their start position (you should decide where it is) without any movement. The first ship that has move control in each screen is always the big ship. Once the user selects a move direction (using the keys, as listed below) the ship will continue to move in this direction even if the user doesn’t press any key, as the ship doesn’t hit a wall or hits a “block” that it cannot move, and as long as the ship was not switched. Once the ship in control is switched, the ship under control is not moving and is waiting for a move direction.

Only one ship can move at a time.

If a ship moves into a wall, it would just stop moving.

If a ship moves into a block, if the block is in the size that the ship can move, and the block can move (i.e. it is not blocked by a wall or by another block) then the ship would move it in the direction of its move, as long as the block is sitting “on a floor” (a floor is just horizontal wall). Once a block is “on the air” (i.e. not “on a floor”) it will start falling.

If a block is pushed to another block, and the size of the two is in the size that the ship can move, it would move the two. However when the first would reach “no floor” it would start to fall, while the other block is still “on the floor”.

Keys:

LEFT	a or A
RIGHT	d or D
UP	w or W
Down	x or X
Switched to the Big Ship	B or b (if we were with the this ship already, just STOP the movement of this ship)
Switched to the Small Ship	S or s (if we were with the this ship already, just STOP the movement of this ship)

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 or 9 to Exit”. But it is not mandatory to present such a message. Do not use the command exit() to exit!!!

When the game is at a pause state, pressing ESC would continue the game, but both ships would be in “STAY” mode (not moving). The ship in control would be the one that was in control before pressing ESC. Pressing 9 when we are in pause mode would exit the game.

As explained above, the screen is consisted of the following information:

- walls (you shall use any reasonable char to draw them)
- a big ship (2*2, use # or any other reasonable char)
- a small ship (2*1, use @ or any other reasonable char)
- “blocks” in positions on screen as you place them - you may want to use different chars or different colors for different “blocks” (possible, not mandatory)
- entry position for each ship (start position)
- exit position (same for both ships)
- total time for this screen

Note that there is one screen in Ex1, which looks the same for all games.
Since there is only one screen, you know how your blocks look, so you can program your code accordingly (i.e. there is no need in Ex1 to create a generic code that would know how to handle all possible block shapes - handle the shapes that you want to support).
You must have in your screen: At least one block that the small ship can move, and needs to move in order to reach the exit! At least one block that the small ship cannot move, but the big ship can move, and needs to move in order for the small ship to reach the exit!

Bonus points: Colors. A smart screen. 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. You MUST provide a readme.txt file that would contain the IDs of the students submitting the exercise. Even if there is only a single submitter (in case you got permission to submit alone) - you still need to put your ID in the readme.txt file.
3. Please indicate inside your readme.txt file the bonus additions that you implemented, if you want to get bonus points for those features.

Exercise 2
In this exercise you will implement the following additions to your game:

Ghosts
Screens may have ghosts. If a ghost hits a ship the player loses (then if there are lives left, the screen restarts).
There are 3 types of Ghosts:
(a) Horizontal (b) Vertical and (c) Wandering
Horizontal Ghosts - move horizontally, left to right and back, on the row where they were born initially (as set by the screen designer)
Vertical Ghosts - move vertically, up and down, on the column where they were born initially (as set by the screen designer)
Wandering Ghosts - wander on screen, according to the algorithm that you would implement, it may have some randomness, but should be totally random.
Note: all ghosts stop at walls! (horizontal and vertical ghosts just turn around on walls, wandering ghosts, well, wander).

Important NOTE:
You can implement in Ex2 only a single type of ghost and add the others in Ex3!
Implementing 3 types of ghosts requires (probably) inheritance and polymorphism - a topic that we will learn in the coming weeks.

Loading Screens from files
The game would look for files in the working directory, with the names *tb*.screen* these files would be loaded in lexicographical order (i.e. *tb01.screen* before *tb02.screen* or *tb_a.screen* before *tb_b.screen* etc.).
You need to submit 3 screens with your exercise!
If there are no files, a proper message would be presented when trying to start a new game.
The menu should have an option to allow running a specific screen, by name.
The screen file should be a text file representing the screen, with:
Header of your choice, the header can hold any required information, such as the amount of time (“ticks”) given for the player to finish this screen. As well as other information that you see as required to be in the header. The header can be separated from the rest of the file with an empty space line, or with any other representation of your choice.
Then the screen itself would appear, with:
@@ - for the position of the small ship (1 and only)
##
- for the position of the big ship (1 and only)
$ - for the initial position of a horizontal ghost (there can be more than one)
! - for the initial position of a vertical ghost (there can be more than one)
% - for the initial position of a wandering ghost (there can be more than one)
W - for walls
& - for the position where the legend 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 ships and any other creature (i.e. surrounded by walls(. You may assume that the screen designer follows this instruction. The size of the actual printed legend shall be not more than 3 lines height * 25 characters.
—
The actual chars on screen for any of the above, can be of your choice.
Note however that the above chars are mandatory, even if you use other chars for drawing the board on screen.
—

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 for the ships and other creatures (as you feel necessary). Name of file shall be in the format corresponding the screen name (e.g. *tb01.steps* for *tb01.screen*)
- There would be a file with the expected result for the screens, in the format corresponding to the screen name: (e.g. *tb01.result* for *tb01.screen*), the file shall include the following information:
    - points of time ("ticks") for this screen where player lost a life (if any)
    - point of time ("ticks") for this screen where the player 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 at least one life for the player till the 3rd screen. You should also add a dedicated readme file called **file_format.txt** explaining your files format, to allow us to understand your file format and create new files or update your 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:

```
thunderbirds.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 when playing from file!).
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 manually 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 would just test 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 would be presented to screen but you may use smaller sleep values than in a regular game.

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

הערה לגבי המונח point in time שמופיע למעלה:
יש לפרש את המילים points in time באופן הבא:
דמיינו שאנו מוסיפים בתחילת ההרצה של כל מסך מונה שמייצג "זמן". כאשר המסך נטען, המונה מאותחל על-מנת שנוכל לתעד את תוצאות הריצה (לבחירתכם - האם לאתחל את המונה באפס ולקדם אותו תוך כדי הריצה על המסך או לאתחלו במקסימום הזמן שיש למסך זה ולהפחית תוך כדי ריצת המסך).
בכל איטרציה של "לולאת המשחק" נעדכן את המונה.
במילים אחרות, המונה מתעדכן בכל "לולאת משחק" (בין אם השחקן הזיז ספינה כלשהי או לא).
כאשר השחקן נפסל, אפשר להחליט לצורך התיעוד בקובץ שמונה הזמן לא יאותחל מחדש (למרות שתצוגת הזמן במסך כן מאותחלת!) או לחילופין אפשר להחליט שמונה הזמן כן יאותחל גם בקובץ.
בכל מקרה, במעבר למסך חדש מונה הזמן יאותחל מחדש.

הכוונה בתרגיל שבתוך קובץ result יש לכתוב את כל ערכי המונה של ה"זמן" שבהם השחקן נפסל במסך זה (אם נפסל), וכן את הערך של מונה ה"זמן" שבו השחקן סיים את המסך (אם סיים). באופן שיאפשר השוואה בין קובץ התוצאות לבין הרצה חדשה של המשחק עם קובץ צעדים (הציפיה היא שקובץ צעדים זהה יוביל תמיד לאותן תוצאות).

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