UA Invaders

PROGRAMMING 1

Adrián Tendero García
Jesús Parra García
Alejandro Benito Marcos

Description

Our project is a recreation of the classic Space Invaders game, developed entirely in C/C++ and in the same way developed using the <gfx.h> library in order to make the graphics of the program, something that is really useful for drawing things such as the enemies, the spacecraft that can be controlled by the player and other features of the game that have been implemented in the best way we could.

On the one hand, we have implemented three different phases with their own levels and each one with its custom difficulty, having every of them three unique levels. As you progress through the game, the levels become harder and it is more tough for the player to advance, although we have ensured that the game is still enjoyable, entertaining an fun to play.

On the other hand, we have managed to create a score system that works perfectly, saving all the results from every execution the player has done, and there can be seen after completing a level or losing. Finally, we have implemented some animations to make the user experience more exciting and improve the visual aspects of the terminal, making it more dynamic.

Implementation

- Libraries *[libraries used indicating what they were used for]*

  <iostream> for I/O instructions

  <gfx.h> to create all the graphics of the game.

  <unistd.h> for creating delay between actions in the game.

  <fstream> for reading, writing, and saving text files .

  <cstring> for using strcpy() function.

Modules [modules that constitute the source code, one table for each module]

Name	*DibEne1*
Task	*Drawing an enemy using the gfx library, in this case the blue martian that shoots bullets.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*DibEne2*
Task	*Drawing an enemy using the gfx library, in this case the classic Space Invaders enemy.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*DibEne3*
Task	*Drawing an enemy using the gfx library, in this case the enemy that is an eye and has two lifes.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*DibEne4*
Task	*Drawing an enemy using the gfx library, in this case the enemy that is an UFO and shoots bullets.*
Input parameters	*Drawing coordinates*

Output parameters	*Does not have*

Name	*DibEne3_1*
Task	*Drawing an enemy using the gfx library, in this case the second skin of the enemy that is an eye.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*dibExp*
Task	*Drawing the explosion, using the gfx library, that appears when an enemy is killed by the player.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*DibP*
Task	*Drawing the character that can be controlled by the player and shoots bullets, in this case it is a spacecraft.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*dibCorazon*
Task	*Drawing a heart, using the gfx library, that represents one life for the player.*
Input parameters	*Drawing coordinates*
Output parameters	*Does not have*

Name	*lvl1_1*
Task	*Creating the first level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl1_2*
Task	*Creating the second level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl1_3*

Task	*Create the third level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl2_1*
Task	*Creating the fourth level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl2_2*
Task	*Creating the fifth level of the game with its own characteristics, including different types of enemies and the position of them.*

Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl2_3*
Task	*Creating the sixth level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl3_1*
Task	*Creating the seventh level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All*

	parameters of the module are passed by reference.
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl3_2*
Task	*Creating the eighth level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition, the quantity of bullets and the input which is the letter Q.*

Name	*lvl3_3*
Task	*Creating the nineth level of the game with its own characteristics, including different types of enemies and the position of them.*
Input parameters	*Player's information, quantity of enemies, the win condition, the quantity of bullets and the last one for reading the keyboard. All parameters of the module are passed by reference.*
Output parameters	*The score obtained, the datatype of the player struct, quantity of enemies, the win condition,*

	the quantity of bullets and the input which is the letter Q.

Name	*DefBala*
Task	*Defining the type of bullet and choosing between the existent ones (enemy 1, enemy 4 or player) that should be used depending on the situation.*
Input parameters	*Quantity of bullets, type of bullet, dimensions of the bullet, identifier of the enemy and the rest of parameters are the datatypes of the structs.*
Output parameters	*Does not have*

Name	*movBalas*
Task	*Changing the position (y) of bullets, if the bullet is of an enemy it sums the position in order to make the bullet go down and if it is a bullet from the player it is a subtraction of the position because we want the bullet to go up.*
Input parameters	*The total number of bullets defined in the struct, the coordinate Y of the window, and the datatypes of the three different structs we have.*
Output parameters	*Does not have*

Name	*dibBala*
Task	*Drawing the bullets and controlling the colour of bullets*
Input parameters	*Quantity of bullets, datatype of the struct that controls the bullets*

Output parameters	*Does not have*

Name	*DibTodasBalas*
Task	*It goes through the array of the bullets and draws them calling the module that draws one module.*
Input parameters	*The total number of bullets defined in the struct and the datatype from the struct of the bullets.*
Output parameters	*Does not have*

Name	*DefEne*
Task	*Defining all the data that has to do with the hitbox of the enemies.*
Input parameters	*Coordinates of the hitbox of the enemies, the rows and columns of the array of enemies, the quantity of enemies and the datatype of the struct of enemy.*
Output parameters	*All input parameters*

Name	*killing*
Task	*It goes bullet by bullet checking all the time if any of the player bullets has hit any of the enemies and if any of the enemies' bullets has hit the player. If a player bullet hits an enemy the bullet disappears defining it as type 0 and that enemy either disappear or if it is the green enemy that has two lives it changes it to its weakened form. If an enemy bullet hits the*

	player, depending on the type of bullet it decreases the player life in 1 or 2.
Input parameters	*The structs of enemies, bullets and player, the parameter hitJ that is a timer for the animation when a bullet hits the player. The variable that is modified to save the score, the matrix of enemies, the identifier of the last enemy of the matrix alive, the total number of bullets, the rows and columns of the enemies' matrix.*
Output parameters	*All the input parameters except the number of bullets and the rows and columns.*

Name	*DetInput*
Task	*Detecting if the key Q of the keyboard has been pressed*
Input parameters	*The value of the key that has been pressed*
Output parameters	*Returning the value of the key that has been pressed by the user*

Name	*accion*
Task	*Controlling the movement of the player (right or left depending on the key pressed), if the player presses the right arrow of the keyboard it moves to the right and if the left arrow is pressed then the ship will move to the left.*
Input parameters	*The key pressed, the coordinate X of the window and the datatype of the player struct.*
Output parameters	*The information of the player, the datatype of the player struct.*

Name	*interfaz*
Task	*Drawing the user interface that appears when you play a level, like the lifes or the borders.*
Input parameters	*The coordinate X and Y of the window and the datatype of the struct of the player.*
Output parameters	*Does not have*

Name	*matriz*
Task	*Reads the enemy matrix and draws the corresponding enemy where we want. It also defines the enemy hitbox. If the enemy is 0 but Contador is >1 it draws the explosion.*
Input parameters	*All necessary variables to draw the enemies and define their hitbox.*
Output parameters	*Does not have.*

Name	*GameOver*
Task	*Checking if the player has lost the game in a level.*
Input parameters	*The rows and columns of the enemies' matrix and the datatypes of the structs of the player and enemy*
Output parameters	*Returns the value "false" in case that the player has lost.*

Name	*countE*

Task	*Counting the enemies left that are in the enemies' matrix and checking if the player has won or not.*
Input parameters	*The matrix of enemies, the rows and the columns of this matrix and the quantity of enemies that are in it.*
Output parameters	*The quantity of enemies that are "alive" in the level and the win value which the function returns.*

Name	*animDrch*
Task	*Using the k parameter, that determines the number of movements that the matrix of enemies must do to reach the opposite edge of the screen considering several factors, this module moves the matrix of enemies from left to right and at the same time implements every module that has to do with the mechanics of the game. Apart from that it detects either if the player has won, if the player has lost or if the player has pressed the quit key "q". If one of these happens it goes immediately out of the loop and the game finish.*
Input parameters	*Score, DatosJugador, DatosEnemigo ,DatosBalas, last enemy, distance, hit times, dimensions, separation, hitbox dimensions, dimensions of the window, amount of enemies, bullet velocity, ticks of the bullet, pixel dimensions, variables of enemies and its bullets, constant, ticks, integer of loss, bool of victory, rows and columns and char to exit and number of bullets and coordinates.*

Output parameters	*Score, DatosJugador, dimensions of the hitbox, coordinates, loss, win, and char to exit.*

Name	*animIzq*
Task	*Using the k parameter, that determines the number of movements that the matrix of enemies must do to reach the opposite edge of the screen considering several factors, this module moves the matrix of enemies from right to left and at the same time implements every module that has to do with the mechanics of the game. Apart from that it detects either if the player has won, if the player has lost or if the player has pressed the quit key "q". If one of these happens it goes immediately out of the loop and the game finish.*
Input parameters	*Score, DatosJugador, DatosEnemigo ,DatosBalas, last enemy, distance, hit times, dimensions, separation, hitbox dimensions, dimensions of the window, amount of enemies, bullet velocity, ticks of the bullet, pixel dimensions, variables of enemies and its bullets, constant, ticks, integer of loss, bool of victory, rows and columns and char to exit and number of bullets and coordinates.*
Output parameters	*Score, DatosJugador, dimensions of the hitbox, coordinates, loss, win, and char to exit.*

Name	*animMatriz*
Task	*Implementing the two previous modules (animDrch and animIzq), and after the call of these modules increments the coordinate Y of the enemies. Also, it checks if the player has*

	won, lost, or if the letter Q of the keyboard has been pressed, in order to stop the animation.
Input parameters	*Score, DatosJugador, DatosEnemigo ,DatosBalas, last enemy, distance, hit times, dimensions, separation, hitbox dimensions, dimensions of the window, amount of enemies, bullet velocity, ticks of the bullet, pixel dimensions, variables of enemies and its bullets, constant, ticks, integer of loss, bool of victory, rows and columns and number of bullets and coordinates.*
Output parameters	*Score, DatosJugador, dimensions of the hitbox, coordinates, loss, win, and char to exit.*

Name	*stats*
Task	*Showing the stats (name, score, lifes left and bullets used) when a level is completed, and when the player dies it also shows the enemies left.*
Input parameters	*Datatype of the player struct, quantity of enemies, win condition and quantity of bullets.*
Output parameters	*Does not have*

Name	*sumScore*
Task	*Every time you kill an enemy you get the corresponding points of that enemy, and it sums all the points obtained in a level.*
Input parameters	*The array of enemies, the rows of the array of enemies and the columns.*
Output parameters	*The score*

Name	*lastE*
Task	*Checking which is the last enemy of the matrix of enemies in a level.*
Input parameters	*The array of enemies, the rows of the array of enemies and the columns.*
Output parameters	*Does not have*

Name	*initialAnimation*
Task	*Module for printing the initial animation of the game.*
Input parameters	*Answer of cin.get()*
Output parameters	*answer*

Name	*mainMenu*
Task	*Module in charge of the interface of the game menu, with all its options.*
Input parameters	*Name of the player, option, DatosJugador datatype, amount of enemies, bool of the victory, number of bullets, char to exit the game and database datatype.*
Output parameters	*DatosJugador datatype, amount of enemies, bool of the victory, number of bullets, char to exit the game.*

Name	*printLetter*
Task	*Module that prints the UA Invaders title.*
Input parameters	*Does not have*

Output parameters	*Does not have*

Name	*credits*
Task	*Display the screen of the credits.*
Input parameters	*Same as mainMenu*
Output parameters	*Same as mainMenu*

Name	*clearbuffer*
Task	*Clear the keyboard buffer of cin.get().*
Input parameters	*Does not have*
Output parameters	*Does not have*

Name	*menuJugar*
Task	*Display the menu to choose the levels and the phases of the game.*
Input parameters	*Same as mainMenu.*
Output parameters	*Same as mainMenu.*

Name	*introAnim*
Task	*Module to display an intro animation to the level.*
Input parameters	*Does not have.*
Output parameters	*Does not have*

Name	*instructions*
Task	*Display the instructions of the game.*

Input parameters	*Answer and player name*
Output parameters	*answer*

Name	*victoryAnim*
Task	*Screen with the game stats in case of win.*
Input parameters	*Answer, player name, score, DatosJugador datatype, total of bullets.*
Output parameters	*Does not have.*

Name	*gameOverAnim*
Task	*Screen with the game stats in case of loss.*
Input parameters	*Answer, player name, score, DatosJugador datatype, amount of enemies left and total bullets*
Output parameters	*Does not have.*

Name	*EPSanimation*
Task	*Animation to exit the program.*
Input parameters	*Does not have.*
Output parameters	*Does not have.*

Name	*saveScore*

Task	*Saves the score of the game in the databases and displays the last scores table.*
Input parameters	*Database datatype, number of level, player name, score, and answer.*
Output parameters	*Does not have.*

Conclusions

The initial objectives of the project have been successfully fulfilled. As a group, we are very proud of our final result, we never thought we would achieve it.

However, during the development of the project, we have encountered many obstacles that we have successfully overcome, such as stack smashing problems, core dumped, bugs with the variables and infinite loops. The most serious error that we have faced was about the cyclic execution of the program, which solution was to reset all of the values of the matrixes and the variables passed by reference. We even had to edit the gfx library itself to create a function programmed by us to close the window from the gfx, as it got bugged and wouldn't close. The movement of the enemies also was a problem to fix.

The project is an extremely good initiative for students to show off their programming skills and demonstrate all they have learnt in the practical sessions. In addition, it teaches the components of the group to teamwork and cooperate in a real project and in something they are motivated to create.

From our point of view, we haven't detected any disadvantages regarding the realization of the project. We consider it is great and should be done in the following years instead of a theoretical or practical exam.

References

Exercises done in the practical sessions, and also YouTube and Internet to search things that we needed to know to develop some mechanics of the game.