IIT BHUBANESWAR

PDS-PROJECT:

CHAITANYA BHARDWAJ – 22EC01004

HET SARSAVA – 22CS01061

RACHIT JAIN – 22EC01006

SHREY JAIN – 22EE01002

CAPSTONE PROJECT

FORZA HORIZON 6

Github link: https://github.com/laughing-philosopher/PDS-Project-Forza-Horizon-6

Introduction

We developed a car game where the player is required to avoid incoming obstacles and the performance is gauged based on a score and the objective is to make it to the high score list. We are calling it *FORZA HORIZON 6* and it is developed completely in C programing language. The libraries used are stdlib.h, graphics.h, conio.h and stdio.h.

IMPORTANT HIGHLIGHTS

- Name of the game is ***FORZA HORIZON 6***.
- We added a loading page that pops us right when you run the code.
- After loading the credits page pops up where the name of each developer is mentioned.
- Then the main meu comes up which displays the controls of the game as well as controls that can be used to navigate and interact.
- By hitting the play button, a 3 second countdown begins after which the actual game starts.
- In the game you have to avoid incoming enemy cars. The enemy car come in three lanes while you can maneuver your car within four lanes, all within premises of the track.
- We have points counting system and after each 1000 points the speed of the game increases.
- The game gets over if you hit an enemy car and if you are one of the top scorers you get to enter your name and your name and score will be visible on the high score menu.
- We also added a customize menu where you can change the theme, style of your car, and also enemy car.
- You can choose between Bugatti, F1 car, Auto, box car, and Spyder for cars.
- For theme, you can choose between Dark Ocean and Gray Mountain.
- When you change theme or car type, a 3 second updating sign pops up and the (current selection) indicator gets updated to you specified selection.

Motivation

Game development is a trendy topic which catches everyone’s attention as a developer. This project is no exception to that trend and involves the usage of C programming language to develop the Capstone Project “Forza Horizon 6”. The development of a game provides immense satisfaction to the developer. During the process of development, we gained thorough knowledge of the functions and logics which were used in the program. Graphics.h is a very popular library which is used for the process of rendering various lines and images on the basis of pixels. We developed keen interest in this library and developed conviction to make a product out of this library.

The learnings developed on the path of development were the constant source of motivation which ensured that we developed the game we visualized as a team during the initial phases of the project. The game makes use of graphics which is of utmost important when developing a game. Graphics add an intense flavor to the game which not only makes the game visually appealing but also ensures that the end users are able to relate to the game in real dimensions and not just through the usage of terminals which involves mental calculations in order to predict the smallest of outcomes.

Our initial idea of developing mathematical problem solver with GUI was adapted into a game when we realized the potential of a game to utilize all the functionalities including mathematics (calculations, coordinates, rendering at specific locations on the basis of the screen size in pixels, etc.). The project involves a large number of functions where each one of us took responsibilities for developing specific functionalities within the game. Since, everyone worked on the project as a team, all the team members were a constant source of inspiration behind the development of the game. As the game saw progress during these last few months, the urge to see the game in its final working condition encouraged the entire group to keep the process moving forward.

Game Overview

The game can be customized by the accordance of the user. The user has the freedom to change the theme of the game between Dark Ocean and Gray Mountain. The user can also customize the shape of the car and enemy cars as well. We have included a variety of designs like Bugatti, F1 car, Box, Auto, and Spider. The game gets over when the user makes contact with an enemy car. The user has the freedom to move in left and right within the premises of the track. This can be achieved by using A and D keys respectively. To navigate in the user interface the user can use W, A, S, and D keys for their respective directions, the user can also use M to return to the main menu and X to return.

Implementation

The user’s car can move in the premises of the track in four lanes while the enemy cars come in three lanes which completely cover the four “My Car” lanes. The speed of the game increases after every one thousand points scored. I you score higher than the top 5 top scorers then you get to enter you name that you want to be displayed on the High Score table.

Development Process

We decided on the topic and then saw some videos on YouTube about how we can approach it. We saw quite a few videos and then we found a video that had a GitHub link in its description. We analyzed the code and decided to make something similar. We took inspiration from it and made our own game with a code of our own. The code contains functions for all the working mechanisms in our game.

Testing and Debugging

At first we faced a lot of errors and problems in our code but we kept analyzing and improving the code until we finally reached a point where the game was user ready, but we didn't stop there and kept improving the code and primarily focused on user experience and simplicity. We kept adding more car designs and ultimately reached the final submission code.

Areas of improvement

Things that we could have improved if we were provided more time and experience:

Firstly, the game could have been made more user-friendly in many ways, for example, we could have allowed the user with more themes in the customize option of the main menu (which currently provides only two options 'Dark Ocean' and 'Gray Mountain'). We could have also provided many more car shape and enemy car shape options in their respective menus. Furthermore, the graphics of the current shapes and cars could have been much more improved (i.e., more details in the structures of cars). The above-mentioned improvements could have been done by being provided more time.

The game currently uses only the “graphics.h” library of C however, if we were provided more time, we could have also used the OpenGL library (or other higher C libraries for Graphics) for better graphics and texture in the game.

The game can also be improved in terms of multimedia usage. Also, the game was developed by us on TurboC which did not support image/audio being added to the output. If we had more experience, we could have used other IDE’s which would have helped us to add better features to the game (such as the image on the loading page of the game or an audio of when the game begins/ends.

Description of Each function:

```
void setgraphics()
```

The function is used to initialize the graphics system by loading passed graphic driver and changing the system into the passed graphic driver. The variable ‘gd’ is a pointer to the graphic driver being used. The term ‘DETECT’ is a macro which tells the compiler to automatically detect the graphic driver that is being used. Also ‘gm’ is a pointer to an integer that specifies the graphics mode to be used. The path is also passed in the function ‘initgraph’ so that the compiler searches in that directory. Basically, this function is the backbone for the usage of graphics in the game.

```
int collisionDetection(struct mycarPosition mc, struct
enemycarPosition ec)
```

This is a simple function that is used to check whether there has been a collision between the enemy car and the player’s own car (or my car). Here with the help of coordinates of the enemy car and my car we check for any collisions.

```
int collisionDetection1( struct mycarPosition mc , struct
enemycarPosition1 ec )
```

This is again a similar simple function that is used to detect for collisions between the enemy car 1 and my car using their coordinates.

void readHighscore()

This function is used to read high score from a file. We used a structure pointer of a file type to declare a file. This function is a clear implementation of file handling functions in C. The file handling functions used are fopen, fclose and fread.

Explanation of the usage of the following functions in brief:

Firstly, we opened highscore.bin (a binary file) only to read (rb mode) using fopen(). If the file did not exist we called the function resetHighscore() else, we read elements from the file using fread() and closed the file using fclose(). Again, if the number of elements read were zero, it means that file did not exist or was not accessible. In such a case we display an error and return to main menu.

void takeuserdetails(long int)

This is an important function that is used to take the user's name. This function is called if the user has scored a high score (that is score higher than the previous 5 highest scores). We have created a character array to store the name of the user. When the game is being played for the first time or the settings have been reset, the name "Player" is displayed on the input box by default else the name of the previous player who had scored a high score is displayed. We used switch(getch()) so as to take input from the user. Using backspace (whose ASCII Corresponds to 8) the user could delete the characters of the previous player's name. Now an important point to note here is that the length of the user's name changes every time. This can create problems, so to avoid any issue, what we have done is that we initialize next character to '0' each time after the user enters a character. Whenever the user enters a character, it is dealt by the default case which stores the character and initializes the next character to '0'. The do – while loop runs until the variable 'the_end' is equal to zero. If the user presses enter / carriage return command / escape, 'the_end' is updated to '1' and thus the do – while loop terminates with the username being stored in the 'input_buf' character array. Then we store the entered name as the previous player's name as

well so that it could be shown as the default name when a higher score is reached. The game high score list is updated.

int updateHighscore();

This function is used to update the high score table. Whenever a high score is created the takeuserdetails() function stores the name of the player at the last position in the high score table. This function places the name of the user and the corresponding high score at the correct position in the high score table by sorting the list according to the highscores. Now we use our file handling functions to open the file "highscore.bin" with the "wb" mode to store the updated high score table in the file. Here we have used the 'fwrite' function. If the file did not exist and number of elements written is zero, we return zero to inform the user that the high score table could not be updated and there has been an error. This could possibly occur if the file is not in the current working directory or in the folder whose path has been specified to the compiler.

int checkifscoreHigh(long int sc)

This is a simple function to check if the score of the current player is greater than the score of the player at the 4^{th} index of the high score table.

void resetsettings()

This is simple function which when called simply displays that the game is being reset and then calls the function resetHighscore() and resetgamedata().

void resetgamedata()

This function sets background, in game text colors and car shape to default values. Also the previous player name is set as default "Player" and then the function updategamedata() is called.

void resetHighscore()

Whenever the game is reset, all the five places in the high score are initialized to zero with corresponding player names being overwritten as default “Player”. This function performs this task and the using file handling function ‘fwrite’ we store the reset values in the file pointed to by fp (that is highscore.bin) and then we close the file.

int updategamedata()

Whenever the game is played the theme and car shape is stored as chosen by the current player so that whenever the game is played next time, the same settings are applied to the game (unless the game has been reset in between).

int readgamedata()

This is an important function which reads all game data such as theme, car shape, previous player name and enemy car shape from a file. This allows us to implement the same settings as that of the previous player in the current game.

int main()

This function is run initially when the game just begins. It is responsible for setting up the game window. It gets the full usable size of the window and calls different functions. It is able to reset the data of the game by calling the resetgamedata() function. It sets up the initial background color with defaultColor() function and calls the loadingscreen() function. Once the loading screen is displayed, credits followed by main menu is displayed on the screen.

void defaultcolor()

This function is for initializing color on the window while the user has not specified any specific color. The default color has been set to blue for the background and green for the text.

void loadingscreen()

This function is the first screen that is displayed on the screen. It shows the name as well as the loading bar on the screen. The loading bar has a time delay of 10 milliseconds between each render and a total of 200 renders are involved in displaying the loading bar which give the appearance of a moving rectangle. The colorSeclect function is also called from this function which sets up the background color and text color according to our preference.

void mainmenu()

This function is used for displaying the main menu on the screen. It displays the name of the game, the controls, and various options such as play, high scores, customize and exit. Whenever a specific button is active, it has a bigger font than the rest of the data to show the currently active button. A selection variable is used to store the current position of the user pointer. When enter is pressed, it passes that selection value and accesses the data stored inside that tab.

void startgame()

This function initially displays a loading screen " Starting in 3…2…1…" to the user. It defines the length of the track and runs 2 enemies simultaneously. When one enemy is rendered, and we reach the end of the loop, the second enemy is activated but with a delay of 5 milliseconds.

During the game, the collision is also detected. If the x coordinate and y coordinate are closeby, the collision is detected and the gameover() function is called.

Struct usage

This game uses a number of structs to create blueprints for different objects whose properties needs to be accessed or used multiple times withing the program. Moreover, whereever we require two objects which have the same types of properties but different data, we have used struct. The enemycarPosition is a struct which contains the coordinates of two edges as parameters.

Mycarposition stores coordinates of the player's car. Highsc stores the name and score of the player who has scored in the top 5. Colors is used for storing the background colors, textcolous for different themes. Previousplayer stores the record of the previous player so that the same

player does not need to input data into the high score field multiple times.

void mycar(mycarLocation)

This function takes the location of the car, defines the center of the car and uses those to draw a car centered on the location provided. We used bar functions and drew shapes to make the car. For different types of cars, we have used functions like floodfill to color a car. We used setfillstyle and setcolor functions to define the color to be filled and the design to be used.

void enemycar(enemycarPosition)/ voidenemycar1(enemycarPosition1)

We used a function which is similar to the mycar function to design enemy cars. The basic principle is the same. We are the random function to randomize the location of enemycars.

void customize()

We have created a customize menu where you can customize the theme of the game, the design of your car, the design of the enemy car, and also, you can reset all settings.

void theme()

In the theme menu, you can select between Gray Mountain and Deep Blue, and your current theme can be identified using the current theme indicator. When you select an option and enter it, Updating text is printed and a “.” is printed three times with one-second delay, and the current theme indicator updates to its new location.

void customizecar()

You get the freedom to select the design of your car irrespective of the design of the enemy car and in the actual game, your car shape gets updated. And like Theme, here as well the current selection indicator indicates your current selection and gets updated after a three-second delay according to your selection.

void customizeenemycar()

You also get the option to select the form of your enemy cars. It is similar to the customizecar function.

void draw (long int sc)

This function is defined for creating a track on which my car and the enemy car will appear. Setcolor is a user-defined function that is used to choose the colour by passing a number as an argument in the function. Numbers 0 to 15 correspond to a particular colour.

The line function takes initial and final coordinates as arguments and prints a line of the specified colour. Sprintf stands for "String Print." Instead of printing on the console, it stores output in a char buffer, which is specified in sprintf.

At specified coordinates in the outtextxy function, the message stored in sprintf is printed. We are printing the score, which is being passed as an argument in the void draw () function.

void pause()

This function allows the user to pause the game in between and resume it according to his or her wish. Line functions are used to draw the pause symbol. A solid bar displays a pause at specified coordinates. The getch() function is present in the conio.h header file. It reads a single character from the keyboard. The switch-case statements work on the keyword input. If P is entered, the game again gets resumed; M opens the main menu; and X exits the game.

void gameover()

This function creates a display screen after the game is over. A solid bar displays "Game Over". The delay() function is used to hold the display screen for a certain amount of time. On entering any keyboard input (any key), the cleardevice() function clears the whole screen, and the sprintf function and outtextxy print the score.

This function also checks whether the score mode is high score or not (in the top 5 high score or not). Accordingly, it prints the message on the

screen. If the score is high, then it calls the takeuserdetails() function to get the details of the high scorer and then return to the main menu.

void showCredits ()

This function displays a screen showing the developers of the game. After the specified delay time, it calls the mainmenu() function.

Text style

All the selections in customize menu have an if-else statement where when you hover over an option, the text gets larger of the particular selection while keeping the text font and orientation same. Also in the main menu.

Contribution

Each and every team member remained equally involved during the development of the project. The work was divided between the team members on the basis of the different functionalities required in the program. The program was divided into:

- **Frontend** which was oriented towards all the functions which displayed data/information to the end user on the graphics terminal. This involved the development of functions – Draw(), pause(), gameover(), display , rendering operations involved in the output on graphics terminal and showcredits(). This part was taken care of by Shrey Jain who ensured that that necessary data can be rendered and stopped at the required time.

- **Backend** which involved the storage of data into a file, accessing the data stored in the file, changing the default colors, initializing the graphics according to the graphics mode and the graphics drive available on the computer. The management of data allowed the previous data to be maintained even after the game has been closed. This involved the usage setgraphics(), setColor(), readgamedata(),updategamedata(),updatehighScore(),resethighScore(), collisionDetection(), managing the score and storing the data if the user has scored in the top five. This part was taken care by Rachit Jain.

- **Functionality** which involved the different functions which displayed the data and interpreted the data according to the data given by the user. The data such as loading screen, main menu, etc which provide the functionalities for managing the user input to redirect it to the required screen and perform the necessary operation. The user uses the data given in the form of WASD characters in order to determine the demand of the user and provide the necessary semantics and outcome. It involved the usage of the functions such as loadingScreen(), startgame(), defaultColor(), main() and different Structs used in the program. This part was taken care of by Chaitanya Bhardwaj.

- **Customize** which involved the setting of the themes, manipulation of the data being rendered on the screen, showing the different shapes for the user car as well as the enemy car. The main menu is able to customize the shape as well as the theme for the user. The themes available are gray mountain and ocean blue. The functions used are mycar(), enemycar(), enemycarCustomized(), themes() and customize(). This part was taken care of by Het Sarsava.

What did you learn from this exercise? Concepts that you explored and implemented outside of your comfort zone.

The things that we learned from this exercise include how to use graphics, how to perform file handling and customization of options given to the user. We learnt how our code can be used in the real world. Team work was also an important learning from this project. Of the so many concepts we implemented in the game, the new concepts that we explored by coming out of our comfort zone include file handling in C and implementation of graphics using the “graphics.h” library. We read many articles / posts / webpages / github codes before sitting down to type the code for this game. We also had to refer books such as “A Book on C” for learning file handling in C.

Conclusion

Overall it was a worthwhile experience developing this game. We learned a lot from this project, we also learned how we can store data from previous iterations and then use it to jump right back to where we left off when the code is run again, also not just coding knowledge but also the thinking process of developing a product as well as how we can tackle errors, and how deriving a systematic procedure can significantly increase productivity and efficiency. Overall, this experience will surely help us in our future endeavors in the field of programming and data structures.