INSIGHTS OF TEAM #2's DEVELOPMENTS FOR PROJECTS THROUGHOUT WEEK 1 TO WEEK 6

IDDI MUNKAILA

8930718

BILAL HUSSAIN

9042866

CONESTOGA COLLEGE

PROFESSOR SAFAT KHAN

PROFESSOR REDA FAYEK

OCTOBER 17TH 2024

Table Of Contents

Muscle & JDM Car Selection Game

Introduction

Team 2, consisting of Bilal and Iddi, embarked on an exciting journey to develop two interactive car selection games as part of our programming project. This project was deemed as their first project as these games were designed to showcase Team 2's understanding of user input handling, loop structures, and the utilization of different C libraries.

Project Overview

The primary objective of our project was to create two distinct code source files, each representing a game revolving around cars. The first game focused on *Muscle* cars and utilized the scanf function for input handling, while the second game centered on *JDM* [Japanese Domestic Market] cars and employed the conio.h library. Both games were required to run continuously using a while loop, request single character user inputs, and print the user-provided input along with its corresponding ASCII integer value.

Code Summary and Functionality:

Muscle Car Game (using scanf):

The first game, developed by Bilal Hussain, focuses on *muscle* cars and utilizes the scanf function for user input. The code structure is as follows:

main.c | Share | Run

```c
#include <stdio.h>
int main() {
char input;
printf("=== Muscle Car Game ===\n");
printf("Press the following keys to choose your favorite muscle car:\n");
printf(" - 'c' for Chevrolet Camaro\n");
printf(" - 'f' for Ford Mustang\n");
printf(" - 'd' for Dodge Charger\n");
printf(" - 'p' for Pontiac GTO\n");
printf(" - 'q' to Quit\n");
printf("=====================================\n");
while (1) {
printf("Please enter your choice: ");
scanf(" %c", &input); // Read a character input
// Handle input for muscle car brands
switch (input) {
case 'c':
printf("You chose Chevrolet Camaro!\n");
break;
case 'f':
printf("You chose Ford Mustang!\n");
break;
case 'd':
printf("You chose Dodge Charger!\n");
break;
case 'p':
printf("You chose Pontiac GTO!\n");
break;
case 'e':
printf("Exiting... Enjoy your ride!\n");
return 0; // Exit the program
default:
printf("Invalid input! Please try again.\n");
break;
}
}
return 0;
}
```

Output

```
/tmp/nd5kPLHxJe.o
=== Muscle Car Game ===
Press the following keys to choose your favorite muscle car:
 - 'c' for Chevrolet Camaro
 - 'f' for Ford Mustang
 - 'd' for Dodge Charger
 - 'p' for Pontiac GTO
 - 'q' to Quit
=====================================
Please enter your choice:
=== Session Ended. Please Run the code again ===
```

The *Muscle* car game runs in a continuous loop, prompting the user to select their favorite *muscle* car by entering a single character. The program then displays the chosen car based on the input using a switch statement for efficient handling of multiple options.

Muscle & JDM Car Selection Game

JDM Car Game (using conio.h): The second game, developed by Iddi Munkaila, focuses on JDM cars and utilizes the conio.h library for input handling. The code structure is as follows:

This game also runs in a continuous loop but uses _kbhit() to check for keypress events and _getch() to capture the input without waiting for Enter. It displays the chosen JDM car based on the user's input and provides the ASCII value of the entered character.

Development Process and Changes

Throughout the development process, we made several changes and improvements to our code:

Theme Selection: Team 2 [Bilal Hussein, Iddi Munkaila] chose car-themed games based on our shared interest in automobiles. This decision was made during our initial brainstorming session, where we recognized the potential to create engaging games around a topic Team 2 [Bilal Hussein, Iddi Munkaila] both enjoyed. Bilal focused on muscle cars, drawing from his passion for American automotive classics, while Iddi concentrated on JDM cars, reflecting his enthusiasm for Japanese performance vehicles.

Compiler Compatibility: Initially, Team 2 [Bilal Hussein, Iddi Munkaila] encountered issues with the conio.h library in VSCode. After thorough research and consultation with our professor, we resolved this by using the Dev-C++ compiler, which successfully ran the code with conio.h. This experience taught Team 2 the importance of compiler compatibility and the need to test our code across different development environments.

Input Handling

Team 2 [Bilal Hussein, Iddi Munkaila] implemented different input handling methods for each game to explore various techniques:

The muscle car game uses scanf, which is straightforward but requires the user to press Enter after each input.
The JDM car game utilizes _kbhit() and _getch() for a more responsive input experience, allowing for immediate character detection without the need for Enter.

Error Handling: Team 2 [Bilal Hussein, Iddi Munkaila] added input validation to both games to handle invalid user inputs and provide appropriate feedback. This enhancement improves the user experience by guiding them towards correct inputs and preventing unexpected behavior.

Muscle & JDM Car Selection Game

ASCII Value Display: As per the project requirements, Team 2 [Bilal Hussein, Iddi Munkaila] included the display of ASCII integer values for each user input in the JDM car game. This feature not only met the project specifications but also provided an educational aspect to the game, allowing users to learn about character encoding.

Collaboration and Time Management

Team 2 [Bilal Hussein, Iddi Munkaila] maintained effective communication throughout the project:

Team 2 [Bilal Hussein, Iddi Munkaila] used Snapchat for quick check-ins and updates, especially during busy periods with other coursework.

Google Meet was utilized for more in-depth discussions and problem-solving sessions.

Team 2 [Bilal Hussein, Iddi Munkaila] respected each other's time and academic commitments, allowing for flexible work schedules while ensuring project progress.

Presentation and Feedback

Team 2 [Bilal Hussein, Iddi Munkaila] presented our project to the class, going first among all the groups. The presentation went well overall, with both the slideshow and code demonstration being successful. However, Team 2 [Bilal Hussein, Iddi Munkaila] received valuable feedback from our professor. These feedback revolved around:

Visual Presentation: Team 2 [Bilal Hussein, Iddi Munkaila] were advised to use a more neutral theme for our presentation slides to improve visibility in various lighting conditions. This feedback highlighted the importance of considering the presentation environment when designing visual aids.

Compiler Compatibility: Our professor emphasized the importance of ensuring our code runs on multiple compilers. While our code worked well in Dev-C++, Team 2 [Bilal Hussein, Iddi Munkaila] recognized the need to write more portable code that can be compiled and run in various environments.

Code Structure: Although not explicitly mentioned, we identified areas whereTeam 2 [Bilal Hussein, Iddi Munkaila] could improve our code structure and organization to enhance readability and maintainability.

Lessons Learned and Future Improvements:

Through this project, Team 2 [Bilal Hussein, Iddi Munkaila] gained valuable insights and identified areas for future improvement:

Cross-platform Development: Team 2 [Bilal Hussein, Iddi Munkaila] learned the importance of writing code that is compatible with multiple compilers and operating systems. In future projects, we will focus on using standard libraries and avoiding platform-specific functions when possible.

User Interface Design: While Team 2 [Bilal Hussein, Iddi Munkaila] games functioned as intended, we recognize the potential to enhance the user interface. Future iterations could include more visual elements or a graphical interface to make the games more engaging.

Code Optimization: As Team 2 [Bilal Hussein, Iddi Munkaila] continue to develop our programming skills, we will explore ways to optimize our code for better performance and efficiency.

Version Control: Although not used extensively in this project, Team 2 [Bilal Hussein, Iddi Munkaila] recognize the benefits of implementing a version control system like Git for better collaboration and code management in future projects.

Conclusion

The development of these two car selection games has been an enlightening and rewarding experience for our team. We successfully implemented the required functionality using different input methods and libraries, meeting all project objectives. The project allowed us to apply our programming knowledge in a practical context, improve our collaboration skills, and gain insights into the software development process.

Moving forward, we are excited to build upon this foundation and incorporate the feedback we received. Our next project will focus on extending the JDM car game using conio.h, implementing a marquee feature with directional controls, and adding more interactive elements. We plan to implement a moving text display that responds to 'L' and 'R' key inputs for left and right movement, generate new marquee lines with other key presses, and include a humorous "BUFFER OVERLOAD!!!" exit message.

JDM Marquee Game/Display

Introduction

Team 2, consisting of Bilal Hussein and Iddi Munkaila, developed the *JDM Marquee Game/Display* as part of their week 3 deliverable for their engineering projects course. This project was deemed as their second project and is an extension of their previous work on JDM (Japanese Domestic Market) car-themed games, incorporating more advanced concepts and interactive elements.

Project Overview

Code Summary: The JDM Marquee Game is a console-based application written in C language. It features a moving marquee text that users can control using keyboard inputs. The code structure includes:

Necessary header files: <stdio.h>, <conio.h>, <string.h>, and <windows.h>
A constant MARQUEE_LENGTH set to 95.
Functions for initialization, user input handling, and marquee display.
A main function that runs the game loop.

Key Functions

Init(): Displays the welcome message and instructions.
getUserInput(): Checks for and returns user input without blocking.
printMarquee(): Renders the marquee text based on the current position.
main(): Contains the game loop and handles user interactions.

Game Controls

'a': Move the marquee left
'd': Move the marquee right
Any other key: Generate a new marquee line
'e': Exit the game

Functionality

The JDM Marquee Game functions as follows:

The game starts by displaying a welcome message and instructions.
A marquee text "Welcome to the JDM Marquee Game! " is displayed and moves across the screen.
Users can control the direction of the marquee using 'a' and 'd' keys.
Pressing any other key generates a new marquee line, displaying the pressed key.
The game continues until the user presses 'e' to exit.
Upon exiting, a humorous message "BUFFER OVERLOAD!!! YOU BROKE THE GAME" is displayed.

JDM Marquee Game/Display

Development Process

Planning and Task Division: On Friday, September 20, 2024, Team 2 [Bilal Hussein, Iddi Munkaila] held a Zoom call to discuss the project plan:

-Team 2 [Iddi Munkaila] decided to create a marquee game based on the conio.h library.
-Team 2 [Bilal Hussein, Iddi Munkaila] agreed on the basic functionality and controls.
-Team 2 [Bilal Hussein] suggested adding a humorous exit message.
-Tasks were divided: Bilal Hussein would handle the presentation template and code testing, while Iddi Munkaila would research new functions and incorporate them into the code.

Implementation

The team worked on implementing the code over the next few days:

-On Monday, September 23rd, Team 2 [Bilal Hussein, Iddi Munkaila] completed the initial version of the code.
-Team 2 [Bilal Hussein, Iddi Munkaila] added all required functions mentioned in the assignment.
-Team 2 [Bilal Hussein, Iddi Munkaila] later modified the code to include the Sleep() function, which they had initially forgotten.

Testing and Refinement

-The code was tested multiple times to ensure proper functionality.
-Team 2 [Bilal Hussein, Iddi Munkaila]made necessary adjustments based on their testing results.

Challenges and Solutions

Time Management: Challenge - Balancing the project with other course loads. Solution -Team 2 [Bilal Hussein, Iddi Munkaila] allocated specific days for focused work on the project and maintained communication through Snapchat and Discord during busier periods.

Function Integration: Challenge - Incorporating all required functions from the assignment. Solution - Team 2 [Bilal Hussein, Iddi Munkaila] conducted thorough research and collaborated to ensure all necessary functions were included in the code.

Code Optimization: Challenge - Ensuring smooth marquee movement and responsive user input. Solution - Team 2 [Bilal Hussein, Iddi Munkaila] implemented non-blocking input using _kbhit() and optimized the marquee rendering function.

Feedback and Improvements

As of the report date, Team 2 [Bilal Hussein, Iddi Munkaila] had not yet presented the project to receive teacher feedback. However, based on previous feedback, they focused on:

- Improving the visual presentation of Team 2 [Bilal Hussein, Iddi Munkaila] slideshow,
-Ensuring code compatibility across different compilers.
-Adding detailed comments to enhance code readability.

JDM Marquee Game/Display

Conclusion

The JDM Marquee Game project allowed Team 2 [Bilal Hussein, Iddi Munkaila] to expand their C programming skills, particularly in handling user input, creating dynamic console displays, and managing game loops. The project demonstrated their ability to collaborate effectively, manage time constraints, and incorporate lessons from previous feedback.

Moving forward, Team 2 plans to:

-Present the project and incorporate any new feedback from their professor.
-Continue refining their C programming skills.
-Explore more advanced topics in game development and user interaction.

Bilal and Iddi's Snake Game - Part 2

Introduction

Team 2, consisting of Bilal Hussein and Iddi Munkaila, developed the Snake Game as part of their engineering project course. This project was deemed as their third project and demonstrated their ability to create a more complex, interactive game using C programming concepts learned throughout the course.

Team 2 which consisted of Bilal and Iddi had two weeks to work on the snake game (Week 5 and Week 6)

Project Overview

Code Summary: Bilal and Iddi's Snake Game is a console-based application written in C. It features a classic snake game where players control a snake to eat food and grow longer. The code structure includes:

Necessary header files: <stdio.h>, <conio.h>, <windows.h>, <stdlib.h>, and <time.h>
Constants for board size and game characters.
Global variables for the game board, snake body, and game state.
Functions for game initialization, user input handling, snake movement, and display.

Key Functions

resetBoard(): Initializes the game board with edges and empty spaces.
spawnFood(): Randomly places food on the board.
draw(): Renders the current game state on the console.
getUserInput(): Handles user input for snake movement and game controls.
moveSnake(): Updates the snake's position and handles collisions and growth.
showLobby(): Displays the game's welcome screen and instructions.

Game Controls

'w': Move snake up
'a': Move snake left
's': Move snake down
'd': Move snake right
'x': Grow snake
'e': Exit game

Functionality

Bilal and Iddi's Snake Game functions as follows:

The game starts with a welcome screen showing instructions.
A snake (represented by '*') appears on a board surrounded by walls ('#').
Food ('^') randomly spawns on the board.
Players control the snake's direction using 'w', 'a', 's', 'd' keys.
The snake grows when it eats food or when 'x' is pressed.
The game continues until the player chooses to exit by pressing 'e'.
The snake can pass through walls, appearing on the opposite side.

Bilal and Iddi's Snake Game - Part 2

Development Process

Planning and Research: On Friday, September 27, 2024, Team 2 began their project:

Team 2 [Bilal Hussein, Iddi Munkaila] recognized the complexity of the assignment but were ready for the challenge.
The team decided to incorporate four function templates suggested by their professor.
Research tasks were divided: Bilal Hussein focused on YouTube videos, while Iddi Munkaila researched articles.

Implementation

Team 2 [Bilal Hussein, Iddi Munkaila] worked on implementing the code over several days:

-On Monday, September 30, Team 2 reviewed their research findings and selected function templates to use.
-Team 2 [Bilal Hussein, Iddi Munkaila] spent 4 hours on a Discord call to complete the initial code.
-The team encountered and resolved several issues, including problems with saving their work.
-Team 2 [Bilal Hussein, Iddi Munkaila] discovered a bug where the snake didn't move as a whole when eating food or growing, but decided to keep it as a challenge.

Documentation and Testing

-On Tuesday, October 1, Team 2 worked on the README file, dividing tasks between game breakdown and controls description.
-By Wednesday, October 2, Team 2 [Bilal Hussein, Iddi Munkaila] had formatted the code and tested it thoroughly, running it 10 times to ensure compliance.

Challenges and Solutions

Code Complexity: Challenge - Creating a more complex game compared to previous projects. Solution - Team 2 utilized function templates provided by their professor and conducted extensive research to understand the required concepts.

Bug Management: Challenge- The snake not moving as a whole when growing. Solution - The team decided to keep this bug as an intentional challenge in the game, demonstrating their ability to make design decisions.

Time Management: Challenge- Balancing the project with other course loads. Solution - Team 2 allocated specific time slots for collaboration and divided tasks efficiently to manage their workload.

Changes and Improvements

Based on their research and professor's suggestions, Team 2 made several improvements to their initial concept:

-Implemented wall-passing functionality for the snake.
-Added a 'grow' button ('x') for additional interactivity.
-Optimized the code structure using provided function templates.
-Created a detailed README file to explain game mechanics and controls.

Bilal and Iddi's Snake Game - Part 2

Feedback Integration

As of the report date, Team 2 had not yet received specific feedback on this project. However, they incorporated lessons from previous assignments:

-Focused on code organization and readability.
-Ensured thorough documentation through the README file.
-Tested the game extensively to catch and address potential issues.

Conclusion

Bilal and Iddi's Snake Game project showcases Team 2's growth in C programming skills, particularly in creating more complex, interactive applications. The project demonstrated their ability to:

-Implement game logic and user interaction in a console environment.
-Manage global state and handle dynamic elements like snake growth and food spawning.
-Work collaboratively on a more challenging project, dividing tasks and integrating their work effectively.

Moving forward, Team 2 plans to:

-Present the project and incorporate any new feedback from their teachers.
-Continue refining their C programming skills, focusing on game development concepts.
-Explore more advanced topics in software development and user interaction design.