National University of Computer and Emerging Sciences

Project Manual

For

Programming Fundamentals Lab

Lab Instructor	Mughees Ismail
Semester	Fall 2023

FAST School of Computer Science

Project Description: John Rambo Console Game

Overview

"John Rambo" is an engaging console-based C++ game that brings the legendary character to life. In this game, players take control of Rambo, guiding him through various obstacles, destroying crates, and combating computer-operated tanks. The gameplay is reminiscent of classic console-based games, where the character moves upwards automatically, giving the illusion of forward movement (sort of like this game https://play.google.com/store/apps/details?id=com.rubysoft.brick&hl=en&gl=US&pli=1)

Group Details

The project is to be attempted in groups of three. Two member groups are allowed after instructor's approval for exceptional scenarios.

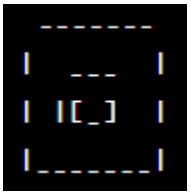
Features

1. **Map Representation:**
    - The game map will be displayed in the console using a 2D character array with a size of 50x30.
    - Empty spaces will be represented by " " (space), obstacles by "|-------|" (hyphens surrounded by vertical bars), and Rambo by characters of your choice, but Rambo must take more than 1 character space.
2. **Obstacle Generation:**
    - Thin obstacles will randomly spawn on the map with a random width between 3 to 7 characters. At any given time, a maximum of 3 obstacles can be present on the screen.
3. **Rambo's Abilities:**
    - Rambo will automatically move upwards, creating the illusion of forward movement.
    - User input will control Rambo's left and right movements using the left and right arrow keys, respectively.
    - Rambo can jump using the space key. When jumping Rambo's character would be changed with a different outlook.

- Pressing 's' will allow Rambo to shoot bullets straight ahead.
- Rambo can only move up down in between the bottom 15 rows.

4. **Obstacle Interaction:**
    - Colliding with thin obstacles results in Rambo's death.
5. **Crates:**

    - Crates will appear on the screen and can be destroyed by Rambo's bullets, for 5 destroyed crates Rambo will gain 1 life.
    - If a crate is not destroyed and Rambo hits a crate, he dies.
6. **Computer-Operated Tanks:**
    - Tanks will follow Rambo horizontally (they will change their positions along the x axis to try and follow Rambo).
    - Tanks can shoot bullets that would disappear after 10 rows, that means that after passing down 10 rows they cannot do any damage and would disappear from the map. You are free to use whatever character(s) to show a tank bullet.
    - Rambo must avoid tank bullets and collisions with tanks.
    - There can be a maximum of 3 tanks in level 1 and 5 in level 2.
    - Tanks cannot collide with each other, once they reach adjacent cells one must move away.



Above image is just a sample. Tanks must take at least 4x4 size in the array.

7. **Distance:**
    - Every step that Rambo takes, he would have covered 2 meters.
    - To achieve 5km Rambo would have to take 2500 steps.
8. **Game Speed:**
    - Adjust the game speed so that it is playable and fun as well.
9. **Game Levels:**
    - The game will have two levels.
    - Increase the game's speed by 1.5x in 2nd level.
    - 2nd level will start once Rambo destroys 10 tanks or has covered 5km in distance, whichever comes first.
10. **High Scores:**
    - Top 10 high scores with names would be stored in a text file.

11. **The Speed Road:**
    - There would be a road in the center of the map, when Rambo comes on top of the game speed would double, once he leaves it the speed would return to normal. Other objects can come on top of the road but it would have no effect on game speed.
    - Objects would be displayed on top of the road, once they leave the road the road would stay as is.

Instructions

Main function should not have any logic inside, all code must be done in functions.

1. Modular Code Structure

- **Function-Centric Approach:**
    - All core logic should be placed inside functions rather than the `main` function.
- **Minimize Code in `main`:**
    - The `main` function should primarily serve as an entry point to the program, containing minimal logic.

2. Function Naming

- **Descriptive Names:**
    - Use clear and descriptive names for functions, variables, and parameters.
    - Favor readability over brevity.

3. Code Reusability

- **Avoid Code Duplication:**
    - If a piece of code is repeating, encapsulate it into a function.

4. Function Responsibilities

- **Single Responsibility Principle:**
    - Each function should have a single responsibility.
    - Divide complex tasks into smaller, manageable functions.

5. Variable Names

- **Meaningful Variables:**
    - Use descriptive variable names that convey their purpose.
    - Avoid single-letter variable names (except for loop counters).

6. Encapsulation

- **Encapsulate Logic:**
    - Group related functionality together within a function.
    - Avoid global variables; use function parameters and return values.

7. Error Handling

- **Error-Handling Functions:**
    - Create separate functions for error handling.
    - Ensure clear messages for debugging purposes.

8. Consistent Indentation and Formatting

- **Maintain Consistency:**
    - Use consistent indentation and formatting throughout the code.
    - Follow a standard style guide (e.g., Google C++ Style Guide).

9. Commenting

- **Comments for Clarity:**
    - Add comments to explain complex logic or functions.
    - Avoid unnecessary comments; write self-explanatory code when possible.

10. Peer Review

- **Peer Review Sessions:**
    - Conduct peer review sessions to ensure code quality and adherence to guidelines.

11. Code Documentation

- **Use Doxygen or Similar Tools:**
    - Consider using documentation generators like Doxygen to automatically generate documentation from code comments.

Helping Content

Possible functions that you might need:

Function Call/Library	Description	Details
system("Color XY")	Print Colored text in C++	https://www.geeksforgeeks.org/how-to-print-colored-text-in-c/
sleep(time_period);	Sleep Function in C++	https://www.geeksforgeeks.org/sleep-function-in-cpp/
_ getch()	Used for reading a single character from the console without echoing it to the screen.	https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/getch-getwch?view=msvc-170
_kbhit()	Checks if a key on the keyboard has been pressed without waiting for the user to hit Enter. It returns a non-zero value if a key has been pressed and zero otherwise.	https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/kbhit?view=msvc-170
#include <SFML/Audio.hpp>	Playing asynchronous sounds	