Introduction to Programming
Task 7.4D: Custom Program Design
BSCP|CS|62|114 Charitha Pieris

The code includes several header files, such as "splashkit.h" and "planet_play.h," which contain necessary function declarations and definitions.

The code defines various data structures using structs, including player_data, coin_data, planet_data, and space_game_data. These structures hold information about the player, coins, planets, and the overall game state.

There are functions defined to create and manipulate different elements of the game. For example, new_player creates a new player with default values, draw_player draws the player on the screen, and handle_input handles keyboard input to control the player's movement.

The new_game function initializes a new game by creating instances of the player, coin, and planet, and setting the count to 0. It returns the initialized space_game_data structure.

The draw_game function clears the screen, draws the player, planet, and coin on the screen, and displays the score. It then refreshes the screen to show the updated display.

The update_game function updates the positions of the coin and planet. If the player collides with the coin, the coin's position is randomized, and the score is increased.

The load_resources function loads the required image resources for the game from a resource bundle file named "planet_play.txt".

The main function is the entry point of the program. It opens a window, loads the resources, creates a new game, and enters a game loop. In each iteration of the loop, it processes events, handles player input, updates the game state, and draws the game on the screen. The loop continues until a quit event is requested.

The code also includes functions related to drawing and updating the planet. The planet_bitmap function returns the appropriate bitmap for a given planet kind, and the draw_planet function draws the planet on the screen. The update_planet function updates the planet's position and wraps it around the screen when it goes off the left edge.

Similarly, there are functions related to the coin, such as new_coin, draw_coin, and update_coin. The new_coin function creates a new coin at a random position, and the update_coin function updates the coin's animation

Overall, this code sets up the game environment, initializes game elements, handles player input, updates the game state, and draws the game on the screen. It provides the basic structure for a simple game where the player controls a spaceship, collects coins, and avoids obstacles represented by planets.

The code uses the following data types:

double: Used to represent floating-point numbers for coordinates and movement speeds.

enum: Used to define enumeration types for ship kinds (ship_kind) and planet kinds (planet_kind). Enumerations provide a set of named values that represent distinct categories or options.

struct: Used to define custom data structures to represent game entities and game state. The struts include player_data (for player information), coin_data (for coin information), planet_data (for planet information), and space_game_data (for overall game state).

bitmap: Represents an image or sprite that can be drawn on the screen.

animation: Represents an animation that can be played and updated.

const: Used to define constant values, such as the movement speed of the planet.

int: Used for integer values, such as the count of collected coins.

These data types are used to store and manipulate different aspects of the game, such as player positions, coin positions, planet types, animation states, and game scores.

The Functions I that use in this program

1. ***new_game():*** Creates a new game by initializing player, coin, planet, and count values.

2. ***update_game(space_game_data& game):*** Updates the positions of the coin and planet and handles collision detection between the player and the coin.

3. ***load_resources():*** Loads the required image resources for the game from a resource bundle

4. ***ship_bitmap(ship_kind kind):*** Returns the appropriate bitmap for a given ship kind (player's ship).

5. ***draw_player(const player_data& player_to_draw):*** Draws the player's ship on the screen.

6. ***handle_input(player_data& player):*** Handles keyboard input to control the player's ship movement.

7. ***new_player():*** Creates a new player with default values.

8. ***new_coin(double x, double y):*** Creates a new coin at the specified position.

9. ***draw_coin(const coin_data& coin):*** Draws the coin on the screen.

10. ***update_coin(coin_data& coin):*** Updates the coin's animation.

These functions are used to initialize, update, and draw various game elements, handle input, and manage the game's overall logic and flow.

The Key functions

draw_game(const space_game_data& game): Draws the player, planet, coin, and score on the screen.

update_game(space_game_data& game): Updates the positions of the coin and planet and handles collision detection between the player and the coin.

handle_input(player_data& player): Handles keyboard input to control the player's ship movement.

These functions are responsible for the core functionality of the game. new_game() initializes the game state, draw_game() handles rendering and displaying game elements, update_game() updates the positions and checks for collisions, handle_input() handles player input, and main() controls the overall flow of the game by managing the game loop and event handling.