FEL PONG

Programmer's manual

File in program:

main.c	The main entry point of the program.
game.c/.h	Main file for playing.
pages.c/.h	Implements the functionality of a game menu and settings. It includes functions for the main menu page, settings page, game start, and game over. The code handles user input, updates the display accordingly, and keeps track of game rounds and player wins. It relies on other files for drawing and utility functions.
checker_game.c/.h	Functions for check different states of the game or objects
draw_game.c/.h	Functions for drawing game board.
drawing.c/.h	All drawing functions.
font_rom8x16.c	File with font for game.
font_types.h	File with font for game.
game_objects.c/.h	Functions for using game objects (racket, ball).
game_effects.c/.h	Functions for working with game's effect (ligtning, snowflake, speed2x) .
mzapo_parlcd.c/.h mzapo_phys.c/.h	Simple program to check LCD functionality on MicroZed based MZ_APO board designed by Petr Porazil at PiKRON
utils.c/.h	Some little functions, witch help for game.

Main.c:

The main entry point of the program.

void set_default_value();

Initializes the default values for game variables and display settings.

Game.c:

int create_round(unsigned char *mem_base, unsigned char *parlcd_mem_base, struct timespec loop_delay, int number_round, int *num_win_round_blue, int *num_win_round_red);

The main function that is called at the beginning of the round. Functions for rendering components and functions that are responsible for the logic of the game are called from it.

@param number_round Number of rounds.

@param num_win_round_blue The number of rounds won by the player on the left.

@param num_win_round_red The number of rounds won by the player on the right.

Checker_game.c:

bool check_rebound_horizntal(int *dx, int *dy, int x_ball, int y_ball, int *score_blue, int *score_red);

This function checks if the ball has reached the horizontal walls. If it hits one of the walls, it modifies the velocity vector as if the ball bounced off the wall. I also add a change in the score in the game.

@param dx The x-coordinate of the ball's velocity vector.

@param dy The y-coordinate of the ball's velocity vector.

@param x_ball Current x coordinate of the ball.

@param y_ball Current y coordinate of the ball.

@param score_blue Player score on the right.

@param score_red Player score on the left.

void check_rebound_vertical(int *dy, int y_ball) ;

If the ball hits a vertical wall, then you need to reflect it.

@param dy The y-coordinate of the ball's velocity vector.

@param y_ball Current y coordinate of the ball.

void check_rebound_rockets(int *dx, int *dy, int x_ball, int y_ball, int y_blue, int y_red, int y_blue_prev, int y_red_prev, int radius_ball);

Here I check if the ball hit the racket. In this case, I consider the velocity vector after reflection.

@param dx The x-coordinate of the ball's velocity vector.

@param dy The y-coordinate of the ball's velocity vector.

@param x_ball Current x coordinate of the ball.

@param y_ball Current y coordinate of the ball.

@param y_blue_prev Previous paddle coordinate value on the right.

@param y_red_prev Previous paddle coordinate value on the left.

@param radius_ball Radius ball.

Drawing.c:

void draw_circles_around(unsigned short color, int activeBut);

Draws circles around buttons based on the active button index.

@param color The color of the circles.

@param activeBut The index of the active button.

void draw_background(unsigned short color);

Draws the background with a specified color.

@param color The color of the background.

void draw_pixel(int x, int y, unsigned short color);

Draws a single pixel with the specified coordinates and color.

The pixel is only drawn if it falls within the screen boundaries.

@param x The x-coordinate of the pixel.

@param y The y-coordinate of the pixel.

@param color The color of the pixel.

void draw_rectangle(int x, int y, int yEnd, int xEnd, unsigned short color);

Draws a rectangle with the specified coordinates, dimensions, and color.

@param x The x-coordinate of the top-left corner of the rectangle.

@param y The y-coordinate of the top-left corner of the rectangle.

@param yEnd The y-coordinate of the bottom-right corner of the rectangle.

@param xEnd The x-coordinate of the bottom-right corner of the rectangle.

@param color The color of the rectangle.

void draw_char(int x, int y, char ch, unsigned short color, int scale);

Draws a character at the specified coordinates with the specified color and scale.

@param x The x-coordinate of the character.

@param y The y-coordinate of the character.

@param ch The character to be drawn.

@param color The color of the character.

@param scale The scale factor for the character size.

void draw_ball(int center_x, int center_y, unsigned short color, int radius);

Draws a filled circle at the specified center coordinates, radius, and color.

@param center_x The x-coordinate of the center of the circle.

@param center_y The y-coordinate of the center of the circle.

@param color The color of the circle.

@param radius The radius of the circle.

void draw_pixel_big(int x, int y, unsigned short color, int scale);

Draws a pixel with a larger size at the specified coordinates and color.

The size of the pixel is determined by the scale factor.

@param x The x-coordinate of the pixel.

@param y The y-coordinate of the pixel.

@param color The color of the pixel.

@param scale The scale factor for the pixel size.

void draw_word_middle(char *word, int x, int y, unsigned short color);

Draws a word with characters centered horizontally at the specified coordinates.

@param word The word to be drawn.

@param x The x-coordinate of the starting position of the word.

@param y The y-coordinate of the starting position of the word.

@param color The color of the word.

void draw_little_word(char *word, int x, int y, unsigned short color);

Draws a word with smaller characters at the specified coordinates.

@param word The word to be drawn.

@param x The x-coordinate of the starting position of the word.

@param y The y-coordinate of the starting position of the word.

@param color The color of the word.

void draw_big_word(char *word, int x, int y, unsigned short color);

Draws a word with larger characters at the specified coordinates.

@param word The word to be drawn.

@param x The x-coordinate of the starting position of the word.

@param y The y-coordinate of the starting position of the word.

@param color The color of the word.

void draw_empty_circle(int center_x, int center_y, int radius, unsigned short color, int scale);

Draws an empty circle at the specified center coordinates, radius, color, and scale.

The circle is only drawn with the outline.

@param center_x The x-coordinate of the center of the circle.

@param center_y The y-coordinate of the center of the circle.

@param radius The radius of the circle.

@param color The color of the circle.

@param scale The scale factor for the circle size.

void draw_active_button_settings(int activeBut);

Draws an active button indicator based on the active button index.

@param activeBut The index of the active button.