API Documentation

ECE 2035

Summer 2016

Badminton

misc.h

Global variables are defined here. Values for these variables cannot be altered during runtime.

U_BUTTON	GND_LVL
R_BUTTON	SKY_COL
D_BUTTON	GND_COL
L_BUTTON	P1_COL
ACC_THRESHOLD	P2_COL
TIME_STEP	SHUT_COL

game.h

```
enum Collision_Codes:
        NO_COLLISION            // No collision detected
        CONT_VOLLEY             // Collision detected; continue volley
        POINT_P1                // Player 1 scores
        POINT_P2                // Player 2 scores
```
The collision codes can be passed and returned as integer parameters and may help to simplify logic.

```
struct racket:
        int player              // PLAYER1 or PLAYER2
        int x, y                // Central X and Y positions of each racket
        int width, height       // Width and height of each racket
        int x0, y0              // Initial positions prior to jump
        float vx0, vy0          // Initial velocities prior to jump
        float time_elapsed      // Cumulative time in air during jump
        bool has_jumped         // Flag for whether the racket has jumped
        int points              // Number of points for a particular racket
```

```
struct shuttle:
        int x0, y0              // Initial positions at beginning of volley
        float vx0, vy0          // Initial velocities at beginning of volley
        int x, y                // Current positions of shuttle
        float time_elapsed      // Cumulative time in air during volley
        bool is_live            // Flag for whether shuttle has been served
        racket *last_possession // Last racket to touch the shuttle
```

void init_racket(racket*, int)

Parameters:

- racket* – pointer to racket to be initialized
- int – player identifier (i.e. PLAYER1 or PLAYER2)

This function initializes a racket with its default values. You may choose to overload this function to initialize variables to other values (i.e. values determined from a game setting).
Hint: Use the '→' operator to access struct members.

void update_racket(racket*)
Parameters:

- racket* – pointer to racket to be updated

This function handles the "running" and "jumping" actions of each racket. Jumping should be governed by the kinematic equations using struct member variables. You may include additional arguments to this function if necessary.

void init_shuttle(shuttle*, racket*)
Parameters:

- shuttle* – pointer to shuttle to be initialized
- racket* – pointer to racket to assign to last_possession field

This function initializes the shuttle with its default values. You may choose to overload this function to initialize variables to other values (i.e. values determined from a game setting).
Hint: Use the '→' operator to access struct members.

int update_shuttle(GSYNC*, shuttle*)
Parameters:

- GSYNC* – pointer to Game_Synchronizer (**Hint:** Useful for reading pixels)
- shuttle* – pointer to shuttle to be updated

Returns: Integer collision code based on updated position of shuttle.
This function handles motion of the shuttle. The shuttle's position should be recalculated for each frame using the kinematic equations.

---

game_synchronizer.h

Game_Synchronizer:

The Game Synchronizer is a struct which coordinates two mbeds to support single or multiplayer gameplay via an Ethernet link. Calls to the various draw commands are queued in an internal buffer. When the GS_update() function is called, the internal buffers are flushed to the LCD, and the graphics primitives are drawn. Each of the drawing functions takes a screen argument. This allows you to draw to SCREEN_P1, SCREEN_P2, or SCREEN_BOTH. **The origin is located at the bottom-left corner of the screen.**

Draw Commands:

void GS_pixel (GSYNC* gs, char screen, int a, int b, int col)

- Draw a pixel of color **col** at location (a, b).

Parameters:

- **GSYNC* gs** – gs is a pointer to the Game_Synchronizer struct.
- **char screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH

- int **a -** x coordinate of the pixel.
- int **b -** y coordinate of the pixel.
- int **col -** The color of the pixel (in 24bpp RGB format).

void GS_background_color (GSYNC* gs, char screen, int color)

- Set the background color for the uLCD to a given color.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **color –** a color in 24bpp format.

void GS_rectangle(GSYNC* gs, char screen, int a, int b, int c, int d, int col)

- Draw a rectangle on the uLCD at the specified coordinates.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **a** – Bottom left corner x coordinate
- int **b** – Bottom left corner y coordinate
- int **c** – Upper right corner x coordinate
- int **d -** Upper right corner y coordinate
- int **col -** The color of the edges of the rectangle.

void GS_filled_rectangle(GSYNC* gs, char screen, int a, int b, int c, int d, int col)

- Draw a filled rectangle with **col** as the fill color.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **a** – Bottom left corner x coordinate
- int **b** – Bottom left corner y coordinate
- int **c** – Upper right corner x coordinate
- int **d -** Upper right corner y coordinate
- int **col** – The fill color.

void GS_circle(GSYNC* gs, char screen, int x , int y , int radius, int color)

- Draw a circle at center (x, y) of specified radius and line color.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **x -** x coordinate of the center of the circle.

- int **y -** y coordinate of the center of the circle.
- int **radius -** The radius of the circle.
- int **color -** The color of the boundary of the circle.

void GS_filled_circle(GSYNC* gs, char screen, int x , int y , int radius, int color)

- Draw a filled circle at center (x, y) of specified radius and fill color.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **x -** x coordinate of the centre of the circle.
- int **y -** y coordinate of the centre of the circle.
- int **radius -** The radius of the circle.
- int **color -** The color of the filled circle.

void GS_triangle(GSYNC* gs, char screen, int a, int b, int c, int d , int e, int f, int col)

- Draw a triangle at the specified 3 points using the **col** as the line color.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **a -** x coordinate for the 1st point of the triangle.
- int **b -** y coordinate for the 1st point of the triangle.
- int **c -** x coordinate for the 2nd point of the triangle.
- int **d -** y coordinate for the 2nd point of the triangle.
- int **e -** x coordinate for the 3rd point of the triangle.
- int **f -** y coordinate for the 3rd point of the triangle.
- int **col -** The color of the triangle.

void GS_line(GSYNC* gs, char screen, int sx, int sy, int ex, int ey, int color)

- Draw a straight line between the points (sx, sy) and (ex, ey) using **color** as the line color.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **sx -** x coordinate for the start point of the line.
- int **sy -** y coordinate for the start point of the line.
- int **ex -** x coordinate for the end point of the line.
- int **ey -** y coordinate for the end point of the line.
- int **color -** The color of the line.

void cls(GSYNC* gs, char screen)

- Clear the screen and place the text cursor at position (0,0) – bottom left of screen.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.

- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH

void GS_locate(GSYNC* gs, char screen, int x, int y)

- Place the text cursor at on the uLCD screen.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **x -** The horizontal ***character index*** from the left, indexed from 0.
- int **y -** The vertical ***character index*** from the top, indexed from 0.

void GS_putc(GSYNC* gs, char screen, char)

- Write a character to the terminal at the text cursor's current position.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- **c -** The character to write to the display.

void GS_textbackground_color (GSYNC* gs, char screen, int col)

- Set the color for the background text to col.

Parameters:

- **GSYNC* gs** – gs is a pointer to the structure Game_Synchronizer.
- char **screen** – screen variable to check SCREEN_P1/SCREEN_P2/SCREEN_BOTH
- int **col** – a color in 24bpp format.

int read_pixel(int x, int y)

- Read the pixel RGB color value at screen location (x, y). The origin is the bottom-left corner of the screen.

Parameters:

- int **x -** x coordinate of the pixel.
- int **y -** y coordinate of the pixel.

Returns:

- int **color -** color in 24bpp format.

int CONVERT_24_TO_16_BPP(int col_24)

- Convert 24 bit-per-pixel (bpp) color format to 16bpp color format.

Parameters:

- int **col_24** – pixel color in 24bpp format.

Returns:

- **int color -** color in 16bpp format.

bool pixel_eq(int color1, int color2)

- Compare two colors and return 1 if they are equal, 0 otherwise. If one color is in 24bpp and the other in 16bpp, the 24bpp color will be converted to 16bpp format for comparison to the other.