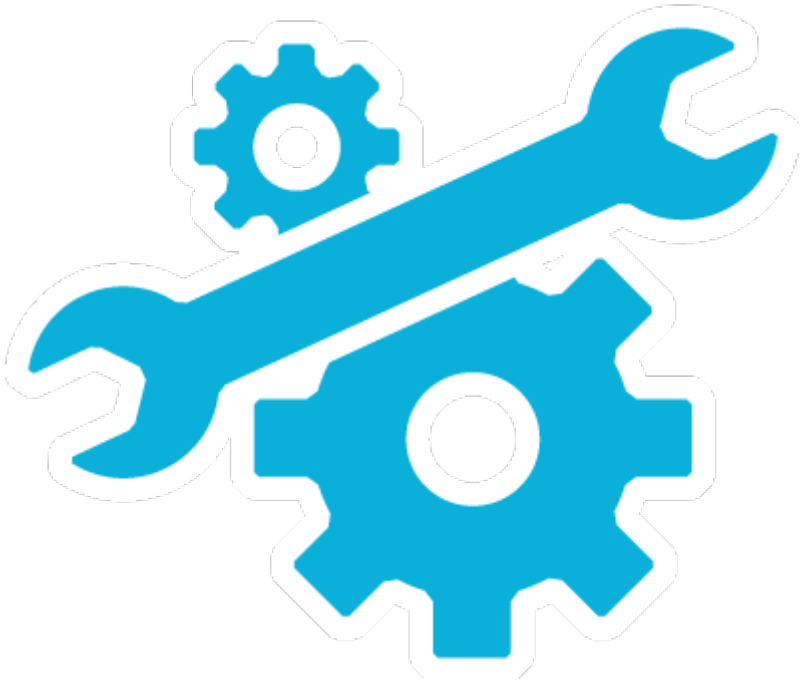
API Documentation



ECE 2035

Spring 2016

Blob Game

blob.h

struct blob variables:

```
float old_x, old_y;   // The old x and y co-ordinates of the blob.
float posx, posy;     // The new x and y co-ordinates of the blob.
float vx, vy;         // The velocity of the blob in the x and y directions.
float rad;            // The radius of the blob.
int color;            // The color of the blob.
bool valid;           // Valid flag for the blob.
bool delete_now;      // d/dt(Valid). Lets us know if we need to "undraw" the blob or not.
```

void BLOB_init(BLOB* b)
Parameters:

- **BLOB* b -** b is a pointer to a blob struct.

Hint: Use the '->' operator to access the structure elements.
This function randomly initializes the position, velocity, and color of a blob. It sets valid to true and delete_now to false.

void BLOB_init(BLOB* b, int rad)
Parameters:

- **BLOB* b -** b is a pointer to a blob struct.
- **int rad -** Radius of the blob.

Hint: Use the '->' operator to access the structure elements.
This function sets everything randomly except for the radius.

void BLOB_init(BLOB* b, int rad, int color)
Parameters:

- **BLOB* b -** b is a structure pointer which is a pointer pointing to struct blob.
- **int rad -** Radius of the blob.
- **int color -** Color of the blob.

Hint: Use the '->' operator to access the structure elements.
This function sets everything randomly except the radius and color.

void BLOB_print(BLOB b)
This function is used to print different attributes of the blob to the pc serial monitor for debugging.
Parameters:

- **BLOB b-** b is a pointer to a blob struct.

float BLOB_dist2(BLOB b1, BLOB b2)
This function is used to calculate the square of the distance between two blobs.
Hint: Use the '**.**' operator to access the structure elements.
Parameters:

- **BLOB b1-** b1 is a variable of data type struct blob used for blob1.
- **BLOB b2-** b2 is a variable of data type struct blob used for blob2.

Returns:

- **float dist-** returns the distance between the blobs as a floating point number.

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. **Note that the game synchronizer places the origin (0,0) at the center 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 position from the left, indexed from 0.
- **int y -** The vertical position from the bottom, 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 (I believe) the center 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.

misc.h

```
// This header file defines useful constants for use in the agar.gt game.

#define U_BUTTON 0
#define R_BUTTON 1
#define D_BUTTON 2
#define L_BUTTON 3


#define ACC_THRESHOLD 0.20

#define WORLD_WIDTH  256
#define WORLD_HEIGHT 256

#define WINNER_P1   0
#define WINNER_P2   1
#define WINNER_TIE  2

#define BGRD_COL   0x0F0066
#define FOOD_COL   0xFFFFFF
#define P1_COL     0xBB0CE8
#define P2_COL     0xFF0030
#define BORDER_COL 0xFFFFFF
```