Frogger

Generated by Doxygen 1.9.3

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

3 Data Structure Documentation

3.1 allegro_t Struct Reference

Data Fields

- ALLEGRO_TIMER * timer
- ALLEGRO_EVENT_QUEUE * queue
- ALLEGRO_DISPLAY * disp
- ALLEGRO_FONT * font
- int font_h
- int font_w
- ALLEGRO_EVENT event
- bool done
- bool redraw

3.1.1 Detailed Description

Definition at line 88 of file allegro_stuff.c.

3.1.2 Field Documentation

3.1.2.1 disp `ALLEGRO_DISPLAY* disp`

Definition at line 97 of file allegro_stuff.c.

3.1.2.2 done `bool done`

Definition at line 108 of file allegro_stuff.c.

3.1.2.3 event `ALLEGRO_EVENT event`

Definition at line 105 of file allegro_stuff.c.

3.1.2.4 font ALLEGRO_FONT* font

Definition at line 100 of file allegro_stuff.c.

3.1.2.5 font_h int font_h

Definition at line 101 of file allegro_stuff.c.

3.1.2.6 font_w int font_w

Definition at line 102 of file allegro_stuff.c.

3.1.2.7 queue ALLEGRO_EVENT_QUEUE* queue

Definition at line 94 of file allegro_stuff.c.

3.1.2.8 redraw bool redraw

Definition at line 110 of file allegro_stuff.c.

3.1.2.9 timer ALLEGRO_TIMER* timer

Definition at line 91 of file allegro_stuff.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/allegro_stuff.c

3.2 car_t Struct Reference

Data Fields

- int x
- int y
- int lane
- int dx
- CAR_TYPE type
- int length
- int count
- bool fast
- bool used

3.2.1 Detailed Description

Definition at line 89 of file entities.c.

3.2.2 Field Documentation

3.2.2.1 count `int count`

Definition at line 97 of file entities.c.

3.2.2.2 dx `int dx`

Definition at line 94 of file entities.c.

3.2.2.3 fast `bool fast`

Definition at line 98 of file entities.c.

3.2.2.4 lane `int lane`

Definition at line 93 of file entities.c.

3.2.2.5 length `int length`

Definition at line 96 of file entities.c.

3.2.2.6 type `CAR_TYPE type`

Definition at line 95 of file entities.c.

3.2.2.7 used `bool used`

Definition at line 99 of file entities.c.

3.2.2.8 x `int x`

Definition at line 91 of file entities.c.

3.2.2.9 y `int y`

Definition at line 92 of file entities.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/entities.c

3.3 coin_t Struct Reference

Data Fields

- int x
- int y
- bool used
-

  struct {
    unsigned int frame_cont
    unsigned int timeout
    unsigned int blink_timer
    unsigned int cont
    bool flag
  } **fx**

3.3.1 Detailed Description

Definition at line 132 of file entities.c.

3.3.2 Field Documentation

3.3.2.1 blink_timer `unsigned int blink_timer`

Definition at line 141 of file entities.c.

3.3.2.2 cont `unsigned int cont`

Definition at line 142 of file entities.c.

3.3.2.3 flag `bool flag`

Definition at line 143 of file entities.c.

3.3.2.4 frame_cont `unsigned int frame_cont`

Definition at line 139 of file entities.c.

3.3.2.5 timeout `unsigned int timeout`

Definition at line 140 of file entities.c.

3.3.2.6 used `bool used`

Definition at line 136 of file entities.c.

3.3.2.7 x `int x`

Definition at line 134 of file entities.c.

3.3.2.8 y `int y`

Definition at line 135 of file entities.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/entities.c

3.4 data_t Struct Reference

Data Fields

- int lives
- unsigned long long score
- unsigned long long score_max
- struct {
  int number
  int time_left
  int time
  long time_ref
  } **run**
- unsigned long frames
- int timer_in_sec
- int difficulty
- char name [MAX_NAME_CHAR]
- unsigned char flag
- bool goals [MAX_GOALS]

3.4.1 Detailed Description

Definition at line 48 of file game_data.c.

3.4.2 Field Documentation

3.4.2.1 difficulty `int difficulty`

Definition at line 65 of file game_data.c.

3.4.2.2 flag `unsigned char flag`

Definition at line 69 of file game_data.c.

3.4.2.3 frames `unsigned long frames`

Definition at line 62 of file game_data.c.

3.4.2.4 goals `bool goals[MAX_GOALS]`

Definition at line 71 of file game_data.c.

3.4.2.5 lives `int lives`

Definition at line 50 of file game_data.c.

3.4.2.6 name `char name[MAX_NAME_CHAR]`

Definition at line 67 of file game_data.c.

3.4.2.7 number `int number`

Definition at line 56 of file game_data.c.

3.4.2.8 score `unsigned long long score`

Definition at line 51 of file game_data.c.

3.4.2.9 score_max `unsigned long long score_max`

Definition at line 52 of file game_data.c.

3.4.2.10 time `int time`

Definition at line 58 of file game_data.c.

3.4.2.11 time_left `int time_left`

Definition at line 57 of file game_data.c.

3.4.2.12 time_ref `long time_ref`

Definition at line 59 of file game_data.c.

3.4.2.13 timer_in_sec `int timer_in_sec`

Definition at line 63 of file game_data.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/game_data.c

3.5 dcoord_t Struct Reference

Data Fields

- uint8_t x
- uint8_t y

3.5.1 Detailed Description

Definition at line 36 of file disdrv.h.

3.5.2 Field Documentation

3.5.2.1 x `uint8_t x`

Definition at line 38 of file disdrv.h.

3.5.2.2 y `uint8_t y`

Definition at line 39 of file disdrv.h.

The documentation for this struct was generated from the following file:

- src/platform/rpi/disdrv.h

3.6 frog_t Struct Reference

Data Fields

- int x
- int y
- int moving
- int facing
- int steps
- unsigned char state
- unsigned char next_action

3.6.1 Detailed Description

Definition at line 77 of file entities.c.

3.6.2 Field Documentation

3.6.2.1 facing `int facing`

Definition at line 82 of file entities.c.

3.6.2.2 moving `int moving`

Definition at line 81 of file entities.c.

3.6.2.3 next_action `unsigned char next_action`

Definition at line 85 of file entities.c.

3.6.2.4 state `unsigned char state`

Definition at line 84 of file entities.c.

3.6.2.5 steps `int steps`

Definition at line 83 of file entities.c.

3.6.2.6 x `int x`

Definition at line 79 of file entities.c.

3.6.2.7 y `int y`

Definition at line 80 of file entities.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/entities.c

3.7 jcoord_t Struct Reference

Data Fields

- int8_t x
- int8_t y

3.7.1 Detailed Description

Definition at line 33 of file joydrv.h.

3.7.2 Field Documentation

3.7.2.1 x `int8_t x`

Definition at line 35 of file joydrv.h.

3.7.2.2 y `int8_t y`

Definition at line 36 of file joydrv.h.

The documentation for this struct was generated from the following file:

- src/platform/rpi/joydrv.h

3.8 log_t Struct Reference

Data Fields

- int x
- int y
- int lane
- int dx
- bool used

3.8.1 Detailed Description

Definition at line 102 of file entities.c.

3.8.2 Field Documentation

3.8.2.1 dx `int dx`

Definition at line 107 of file entities.c.

3.8.2.2 lane `int lane`

Definition at line 106 of file entities.c.

3.8.2.3 used `bool used`

Definition at line 108 of file entities.c.

3.8.2.4 x `int x`

Definition at line 104 of file entities.c.

3.8.2.5 y `int y`

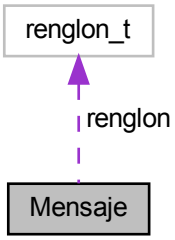
Definition at line 105 of file entities.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/entities.c

3.9 Mensaje Struct Reference

Collaboration diagram for Mensaje:



Data Fields

- char msj [L_MAX]
- int posicion
- int index
- int longitud
- int j
- bool habilitacion
- bool mover_texto
- bool repetir_msj
- renglon_t renglon

3.9.1 Detailed Description

Definition at line 44 of file mensajes.h.

3.9.2 Field Documentation

3.9.2.1 habilitacion `bool habilitacion`

Definition at line 51 of file mensajes.h.

3.9.2.2 index `int index`

Definition at line 48 of file mensajes.h.

3.9.2.3 j `int j`

Definition at line 50 of file mensajes.h.

3.9.2.4 longitud `int longitud`

Definition at line 49 of file mensajes.h.

3.9.2.5 mover_texto `bool mover_texto`

Definition at line 52 of file mensajes.h.

3.9.2.6 msj `char msj[L_MAX]`

Definition at line 46 of file mensajes.h.

3.9.2.7 posicion `int posicion`

Definition at line 47 of file mensajes.h.

3.9.2.8 renglon `renglon_t renglon`

Definition at line 54 of file mensajes.h.

3.9.2.9 repetir_msj `bool repetir_msj`

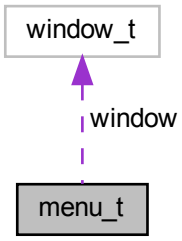
Definition at line 53 of file mensajes.h.

The documentation for this struct was generated from the following file:

- src/platform/rpi/mensajes.h

3.10 menu_t Struct Reference

Collaboration diagram for menu_t:



Data Fields

- window_t window [MENU_WINDOW_MAX]
- int actual_window

3.10.1 Detailed Description

Definition at line 41 of file menu.c.

3.10.2 Field Documentation

3.10.2.1 actual_window `int actual_window`

Definition at line 45 of file menu.c.

3.10.2.2 window `window_t window[MENU_WINDOW_MAX]`

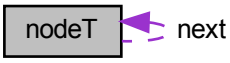
Definition at line 43 of file menu.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/menu.c

3.11 nodeT Struct Reference

Collaboration diagram for nodeT:



Data Fields

- event_t data
- struct nodeT * next

3.11.1 Detailed Description

Definition at line 25 of file queue.c.

3.11.2 Field Documentation

3.11.2.1 data `event_t data`

Definition at line 27 of file queue.c.

3.11.2.2 next `struct nodeT* next`

Definition at line 28 of file queue.c.

The documentation for this struct was generated from the following file:

- src/queue.c

3.12 pair_xy_t Struct Reference

Data Fields

- int x
- int y

3.12.1 Detailed Description

Definition at line 135 of file geometry.h.

3.12.2 Field Documentation

3.12.2.1 x `int x`

Definition at line 137 of file geometry.h.

3.12.2.2 y `int y`

Definition at line 138 of file geometry.h.

The documentation for this struct was generated from the following file:

- src/platform/pc/geometry.h

3.13 renglon_t Union Reference

Data Fields

- uint32_t completo
-

```
struct {
    uint16_t mitad_der
    uint16_t mitad_izq
};
```

3.13.1 Detailed Description

Definition at line 34 of file mensajes.h.

3.13.2 Field Documentation

3.13.2.1 completo `uint32_t completo`

Definition at line 36 of file mensajes.h.

3.13.2.2 mitad_der `uint16_t mitad_der`

Definition at line 39 of file mensajes.h.

3.13.2.3 mitad_izq `uint16_t mitad_izq`

Definition at line 40 of file mensajes.h.

The documentation for this union was generated from the following file:

- src/platform/rpi/mensajes.h

3.14 sounds_t Struct Reference

Data Fields

- ALLEGRO_AUDIO_STREAM * stream
- unsigned char stream_state
- struct {
  ALLEGRO_SAMPLE * jump
  ALLEGRO_SAMPLE * crash
  ALLEGRO_SAMPLE * goal
  ALLEGRO_SAMPLE * low_time
  ALLEGRO_SAMPLE * click
  ALLEGRO_SAMPLE * bonus
  ALLEGRO_SAMPLE * run_completed
  ALLEGRO_SAMPLE * drowned
  ALLEGRO_SAMPLE * menu_enter
  ALLEGRO_SAMPLE * new_max_score
  ALLEGRO_SAMPLE * exiting
  ALLEGRO_SAMPLE * no_time
  ALLEGRO_SAMPLE * coin_drop
  } **samples**

3.14.1 Detailed Description

Definition at line 114 of file allegro_stuff.c.

3.14.2 Field Documentation

3.14.2.1 bonus `ALLEGRO_SAMPLE* bonus`

Definition at line 126 of file allegro_stuff.c.

3.14.2.2 click `ALLEGRO_SAMPLE* click`

Definition at line 125 of file allegro_stuff.c.

3.14.2.3 coin_drop `ALLEGRO_SAMPLE* coin_drop`

Definition at line 133 of file allegro_stuff.c.

3.14.2.4 crash `ALLEGRO_SAMPLE* crash`

Definition at line 122 of file allegro_stuff.c.

3.14.2.5 drowned `ALLEGRO_SAMPLE* drowned`

Definition at line 128 of file allegro_stuff.c.

3.14.2.6 exiting `ALLEGRO_SAMPLE* exiting`

Definition at line 131 of file allegro_stuff.c.

3.14.2.7 goal `ALLEGRO_SAMPLE* goal`

Definition at line 123 of file allegro_stuff.c.

3.14.2.8 jump `ALLEGRO_SAMPLE* jump`

Definition at line 121 of file allegro_stuff.c.

3.14.2.9 low_time `ALLEGRO_SAMPLE* low_time`

Definition at line 124 of file allegro_stuff.c.

3.14.2.10 menu_enter `ALLEGRO_SAMPLE* menu_enter`

Definition at line 129 of file allegro_stuff.c.

3.14.2.11 new_max_score `ALLEGRO_SAMPLE* new_max_score`

Definition at line 130 of file allegro_stuff.c.

3.14.2.12 no_time `ALLEGRO_SAMPLE* no_time`

Definition at line 132 of file allegro_stuff.c.

3.14.2.13 run_completed `ALLEGRO_SAMPLE* run_completed`

Definition at line 127 of file allegro_stuff.c.

3.14.2.14 stream `ALLEGRO_AUDIO_STREAM* stream`

Definition at line 116 of file allegro_stuff.c.

3.14.2.15 stream_state `unsigned char stream_state`

Definition at line 117 of file allegro_stuff.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/allegro_stuff.c

3.15 sprites_menu_t Struct Reference

3.15.1 Detailed Description

Definition at line 50 of file allegro_stuff.h.

The documentation for this struct was generated from the following file:

- src/platform/pc/allegro_stuff.h

3.16 sprites_t Struct Reference

Estructura principal de spritesheets.

`#include <allegro_stuff.h>`

Data Fields

- ALLEGRO_BITMAP * frog_uncut
- ALLEGRO_BITMAP * frog [8]
- ALLEGRO_BITMAP * background
- ALLEGRO_BITMAP * log
- ALLEGRO_BITMAP * cars_uncut
- ALLEGRO_BITMAP * car [CAR_TYPE_N]
- ALLEGRO_BITMAP * turtle_uncut
- ALLEGRO_BITMAP * turtle [TURTLE_FRAMES]
- ALLEGRO_BITMAP * heart
- struct {
  ALLEGRO_BITMAP * uncut
  ALLEGRO_BITMAP * option [MENU_STATE_MAX]
  ALLEGRO_BITMAP * background
  } **menu** [MENU_WINDOW_MAX]
- ALLEGRO_BITMAP * credits
- ALLEGRO_BITMAP * name
- ALLEGRO_BITMAP * icon
- ALLEGRO_BITMAP * dead
- struct {
  ALLEGRO_BITMAP * uncut
  ALLEGRO_BITMAP * frame [SPRITE_COIN_FRAMES]
  } **coin**
- struct {
  ALLEGRO_BITMAP * uncut
  ALLEGRO_BITMAP * frame [SPRITE_SPLASH_FRAMES]
  } **splash**
- ALLEGRO_BITMAP * border

3.16.1 Detailed Description

Estructura principal de spritesheets.

Definition at line 59 of file allegro_stuff.h.

3.16.2 Field Documentation

3.16.2.1 background `ALLEGRO_BITMAP* background`

Definition at line 64 of file allegro_stuff.h.

3.16.2.2 border `ALLEGRO_BITMAP* border`

Definition at line 103 of file allegro_stuff.h.

3.16.2.3 car `ALLEGRO_BITMAP* car[CAR_TYPE_N]`

Definition at line 69 of file allegro_stuff.h.

3.16.2.4 cars_uncut `ALLEGRO_BITMAP* cars_uncut`

Definition at line 68 of file allegro_stuff.h.

3.16.2.5 credits `ALLEGRO_BITMAP* credits`

Definition at line 83 of file allegro_stuff.h.

3.16.2.6 dead `ALLEGRO_BITMAP* dead`

Definition at line 89 of file allegro_stuff.h.

3.16.2.7 frame `ALLEGRO_BITMAP* frame[SPRITE_SPLASH_FRAMES]`

Definition at line 94 of file allegro_stuff.h.

3.16.2.8 frog `ALLEGRO_BITMAP* frog[8]`

Definition at line 62 of file allegro_stuff.h.

3.16.2.9 frog_uncut `ALLEGRO_BITMAP* frog_uncut`

Definition at line 61 of file allegro_stuff.h.

3.16.2.10 heart `ALLEGRO_BITMAP* heart`

Definition at line 74 of file allegro_stuff.h.

3.16.2.11 icon `ALLEGRO_BITMAP* icon`

Definition at line 87 of file allegro_stuff.h.

3.16.2.12 log `ALLEGRO_BITMAP* log`

Definition at line 66 of file allegro_stuff.h.

3.16.2.13 name `ALLEGRO_BITMAP* name`

Definition at line 85 of file allegro_stuff.h.

3.16.2.14 option `ALLEGRO_BITMAP* option[MENU_STATE_MAX]`

Definition at line 79 of file allegro_stuff.h.

3.16.2.15 turtle `ALLEGRO_BITMAP* turtle[TURTLE_FRAMES]`

Definition at line 72 of file allegro_stuff.h.

3.16.2.16 turtle_uncut `ALLEGRO_BITMAP* turtle_uncut`

Definition at line 71 of file allegro_stuff.h.

3.16.2.17 uncut `ALLEGRO_BITMAP* uncut`

Definition at line 78 of file allegro_stuff.h.

The documentation for this struct was generated from the following file:

- src/platform/pc/allegro_stuff.h

3.17 state_diagram_edge Struct Reference

Collaboration diagram for state_diagram_edge:

Data Fields

- event_t evento
- STATE * proximo_estado
- void(* p_rut_accion)(void)

3.17.1 Detailed Description

Definition at line 53 of file fsm.c.

3.17.2 Field Documentation

3.17.2.1 evento `event_t evento`

Definition at line 55 of file fsm.c.

3.17.2.2 p_rut_accion `void(* p_rut_accion) (void)`

Definition at line 57 of file fsm.c.

3.17.2.3 proximo_estado `STATE* proximo_estado`

Definition at line 56 of file fsm.c.

The documentation for this struct was generated from the following file:

- src/fsm.c

3.18 turtle_pack_t Struct Reference

Data Fields

- int x
- int y
- int lane
- int dx
- bool used
- unsigned char turtles_in_pack
- struct {
  unsigned char frame
  unsigned int timeout
  unsigned int cont
  } **fx**
- int wide
- unsigned char state

3.18.1 Detailed Description

Definition at line 112 of file entities.c.

3.18.2 Field Documentation

3.18.2.1 cont `unsigned int cont`

Definition at line 125 of file entities.c.

3.18.2.2 dx `int dx`

Definition at line 117 of file entities.c.

3.18.2.3 frame `unsigned char frame`

Definition at line 123 of file entities.c.

3.18.2.4 lane `int lane`

Definition at line 116 of file entities.c.

3.18.2.5 state `unsigned char state`

Definition at line 129 of file entities.c.

3.18.2.6 timeout `unsigned int timeout`

Definition at line 124 of file entities.c.

3.18.2.7 turtles_in_pack `unsigned char turtles_in_pack`

Definition at line 119 of file entities.c.

3.18.2.8 used `bool used`

Definition at line 118 of file entities.c.

3.18.2.9 wide `int wide`

Definition at line 128 of file entities.c.

3.18.2.10 x int x

Definition at line 114 of file entities.c.

3.18.2.11 y int y

Definition at line 115 of file entities.c.

The documentation for this struct was generated from the following file:

- src/platform/pc/entities.c

3.19 window_t Struct Reference

Data Fields

- int actual_state
- int max_states

3.19.1 Detailed Description

Definition at line 35 of file menu.c.

3.19.2 Field Documentation

3.19.2.1 actual_state int actual_state

Definition at line 37 of file menu.c.

3.19.2.2 max_states int max_states

Definition at line 38 of file menu.c.

The documentation for this struct was generated from the following file:

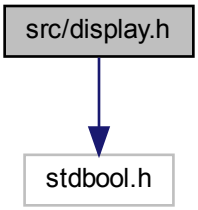
- src/platform/pc/menu.c

4 File Documentation

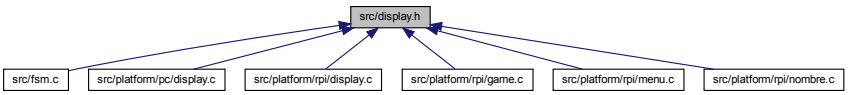
4.1 src/display.h File Reference

Header del modulo display Vinculo entre la fsm y las plataformas en lo que respecta a la visualizacion del juego.

`#include <stdbool.h>`
Include dependency graph for display.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum **posiciones_mensajes** {
  POS_MSJ_MENU , **POS_MSJ_DIFICULTAD** , **POS_MSJ_RANKING** , **POS_MSJ_NOMBRE** ,
  POS_MSJ_PASAR , **POS_MSJ_PAUSA** , **POS_MSJ_NEW_HI_SCORE** , **POS_MSJ_GAME_OVER** ,
  POS_OPCION , **POS_RANKING_2** , **POS_CREDITOS** }

Functions

- bool iniciarDisplay ()
  Inicializa el display de la plataforma.
- void actualizarDisplay ()
  Actualiza el display de la plataforma.
- void limpiarDisplay ()
  Limpia el display de la plataforma.
- void mostrarTexto (char ∗txt, int pos)
  Muestra un texto dado en una posicion dada (retiene el flujo)
- void dejarTexto (char ∗txt, int pos, bool repetir)

Deja el texto en la posición data (no retiene)

- void cargarRanking (void)

  Inicia muestreo de ranking en la plataforma.
- void mostrarRanking (void)

  Bucle que muestra el ranking. Finaliza desde dentro, y hace que finalice el thread de ranking.
- void cargarCreditos (void)

  Inicializa los cretidos en la plataforma.
- void mostrarCreditos (void)

  Bucle que muestra los creditos. Finaliza desde dentro, y hace que finalice el thread de ranking.
- void reconfigurarDisplayON (void)

  Reconfigura el display de la plataforma y lo habilita.
- void reconfigurarDisplayOFF (void)

  Reconfigura el display de la plataforma y lo deshabilita.

4.1.1 Detailed Description

Header del modulo display Vinculo entre la fsm y las plataformas en lo que respecta a la visualizacion del juego.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file display.h.

4.1.2 Enumeration Type Documentation

4.1.2.1 posiciones_mensajes `enum posiciones_mensajes`

Definition at line 27 of file display.h.

4.1.3 Function Documentation

4.1.3.1 actualizarDisplay() `void actualizarDisplay ( )`

Actualiza el display de la plataforma.

Definition at line 58 of file display.c.

4.1.3.2 cargarCreditos() `void cargarCreditos ( void )`

Inicializa los cretidos en la plataforma.

Definition at line 135 of file display.c.

4.1.3.3 cargarRanking() `void cargarRanking ( void )`

Inicia muestreo de ranking en la plataforma.

Parameters

txt	

Definition at line 74 of file display.c.

4.1.3.4 dejarTexto() `void dejarTexto ( char * txt, int pos, bool repetir )`

Deja el texto en la posición data (no retiene)

Parameters

txt	
pos	
repetir	

Definition at line 70 of file display.c.

4.1.3.5 iniciarDisplay() `bool iniciarDisplay ( )`

Inicializa el display de la plataforma.

Returns

true Exito

false Error

Definition at line 48 of file display.c.

4.1.3.6 limpiarDisplay() `void limpiarDisplay ( )`

Limpia el display de la plataforma.

Definition at line 62 of file display.c.

4.1.3.7 mostrarCreditos() `void mostrarCreditos ( void )`

Bucle que muestra los creditos. Finaliza desde dentro, y hace que finalice el thread de ranking.

Returns

true No finaliz

false Finaliza

Definition at line 140 of file display.c.

4.1.3.8 mostrarRanking() `void mostrarRanking ( void )`

Bucle que muestra el ranking. Finaliza desde dentro, y hace que finalice el thread de ranking.

Definition at line 131 of file display.c.

4.1.3.9 mostrarTexto() `void mostrarTexto ( char * txt, int pos )`

Muestra un texto dado en una posicion dada (retiene el flujo)

Parameters

txt	Texto
pos	Posicion

Definition at line 66 of file display.c.

4.1.3.10 reconfigurarDisplayOFF() `void reconfigurarDisplayOFF ( void )`

Reconfigura el display de la plataforma y lo deshabilita.

Definition at line 162 of file display.c.

4.1.3.11 reconfigurarDisplayON() void reconfigurarDisplayON (void)

Reconfigura el display de la plataforma y lo habilita.

Definition at line 157 of file display.c.

4.2 display.h

Go to the documentation of this file.

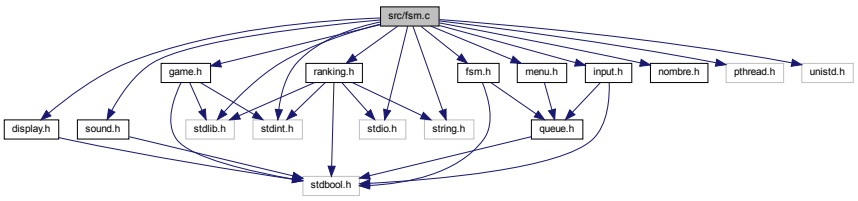
```
00001
00013 #ifndef _DISPLAY_H_
00014 #define _DISPLAY_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdbool.h>
00021
00022 /*****************************************************************************
00023  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00024  *****************************************************************************/
00025
00026 // Posiciones de mensajes
00027 enum posiciones_mensajes
00028 {
00029     POS_MSJ_MENU,
00030     POS_MSJ_DIFICULTAD,
00031     POS_MSJ_RANKING,
00032     POS_MSJ_NOMBRE,
00033     POS_MSJ_PASAR,
00034     POS_MSJ_PAUSA,
00035     POS_MSJ_NEW_HI_SCORE,
00036     POS_MSJ_GAME_OVER,
00037     POS_OPCION,
00038     POS_RANKING_2,
00039     POS_CREDITOS
00040 };
00041
00042 /*****************************************************************************
00043  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00044  *****************************************************************************/
00045
00052 bool iniciarDisplay();
00053
00058 void actualizarDisplay();
00059
00064 void limpiarDisplay();
00065
00072 void mostrarTexto(char *txt, int pos);
00073
00081 void dejarTexto(char *txt, int pos, bool repetir);
00082
00088 void cargarRanking(void);
00089
00094 void mostrarRanking(void);
00095
00100 void cargarCreditos(void);
00101
00108 void mostrarCreditos(void);
00109
00114 void reconfigurarDisplayON(void);
00115
00120 void reconfigurarDisplayOFF(void);
00121
00122 /*****************************************************************************
00123  *****************************************************************************/
00124
00125 #endif // _DISPLAY_H_
```

4.3 src/fsm.c File Reference

Source del modulo fsm. Administra la máquina de estados, siendo el engine del juego.

```
#include "fsm.h"
#include "display.h"
#include "game.h"
#include "menu.h"
#include "input.h"
#include "nombre.h"
#include "sound.h"
#include "ranking.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>
```

Include dependency graph for fsm.c:



Data Structures

- struct state_diagram_edge

Macros

- #define FIN_TABLA 0xFF
- #define CTE_OPCION 100
- #define FIX_CPU_USAGE_SLEEP_US 500

Typedefs

- typedef struct state_diagram_edge STATE

Functions

- bool inicializarFsm (void)

  Inicializa la FSM, notificando si tuvo exito.
- void fsm (event_t evento_actual)

  Intérprete de la FSM. Se encarga de hacer correr la FSM a partir de un estado y evento dados.
- void fixHighCpuUsage (void)

  Fixea consumo elevado de cpu en el while loop principal.

Variables

- STATE en_menu_ppal [ ]
- STATE menu_ppal_esperando_opcion [ ]
- STATE seleccionando_dificultad [ ]
- STATE viendo_ranking [ ]
- STATE viendo_creditos [ ]
- STATE poniendo_nombre [ ]
- STATE jugando [ ]
- STATE en_pausa [ ]
- STATE en_pausa_esperando_opcion [ ]
- STATE en_game_over [ ]
- STATE en_game_over_esperando_opcion [ ]

4.3.1 Detailed Description

Source del modulo fsm. Administra la máquina de estados, siendo el engine del juego.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file fsm.c.

4.3.2 Macro Definition Documentation

4.3.2.1 CTE_OPCION `#define CTE_OPCION 100`

Definition at line 41 of file fsm.c.

4.3.2.2 FIN_TABLA `#define FIN_TABLA 0xFF`

Definition at line 38 of file fsm.c.

4.3.2.3 FIX_CPU_USAGE_SLEEP_US `#define FIX_CPU_USAGE_SLEEP_US 500`

Definition at line 44 of file fsm.c.

4.3.3 Typedef Documentation

4.3.3.1 STATE `typedef struct state_diagram_edge STATE`

Definition at line 50 of file fsm.c.

4.3.4 Function Documentation

4.3.4.1 fixHighCpuUsage() `void fixHighCpuUsage ( void )`

Fixea consumo elevado de cpu en el while loop principal.

Not the best solucion, but sirve...

https://softwareengineering.stackexchange.com/questions/256524/infinite-while-loop-cpu-

Definition at line 354 of file fsm.c.

4.3.4.2 fsm() `void fsm ( event_t evento_actual )`

Intérprete de la FSM. Se encarga de hacer correr la FSM a partir de un estado y evento dados.

Parameters

p_tabla_estado	Estado actual.
evento_actual	Evento recibido.

Definition at line 336 of file fsm.c.

4.3.4.3 inicializarFsm() `bool inicializarFsm ( void )`

Inicializa la FSM, notificando si tuvo exito.

Returns

true Exito

false Error

Definition at line 316 of file fsm.c.

4.3.5 Variable Documentation

4.3.5.1 en_game_over STATE en_game_over

Initial value:

```
=
    {
        {ENTER, en_game_over_esperando_opcion, procesar_enter_menu},
        {ARRIBA, en_game_over, subirOpcion},
        {ABAJO, en_game_over, bajarOpcion},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, en_game_over, do_nothing}}
```

Definition at line 295 of file fsm.c.

4.3.5.2 en_game_over_esperando_opcion STATE en_game_over_esperando_opcion

Initial value:

```
=
    {
        {CTE_OPCION, jugando, iniciar_juego},
        {CTE_OPCION + 1, en_menu_ppal, ir_a_menu_ppal},
        {FIN_TABLA, en_game_over_esperando_opcion, do_nothing}}
```

Definition at line 303 of file fsm.c.

4.3.5.3 en_menu_ppal STATE en_menu_ppal

Initial value:

```
=
    {
        {ENTER, menu_ppal_esperando_opcion, procesar_enter_menu},
        {ARRIBA, en_menu_ppal, subirOpcion},
        {ABAJO, en_menu_ppal, bajarOpcion},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, en_menu_ppal, do_nothing}}
```

Definition at line 219 of file fsm.c.

4.3.5.4 en_pausa STATE en_pausa

Initial value:

```
=
    {
        {ENTER, en_pausa_esperando_opcion, procesar_enter_menu},
        {ARRIBA, en_pausa, subirOpcion},
        {ABAJO, en_pausa, bajarOpcion},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, en_pausa, do_nothing}}
```

Definition at line 280 of file fsm.c.

4.3.5.5 en_pausa_esperando_opcion STATE en_pausa_esperando_opcion

Initial value:

```
=
    {
        {CTE_OPCION, jugando, continuar},
        {CTE_OPCION + 1, jugando, iniciar_juego},
        {CTE_OPCION + 2, en_menu_ppal, ir_a_menu_ppal},
        {FIN_TABLA, en_pausa_esperando_opcion, do_nothing}}
```

Definition at line 288 of file fsm.c.

4.3.5.6 jugando STATE jugando

Initial value:

```
=
    {
        {ENTER, en_pausa, pausar},
        {GAME_OVER, en_game_over, procesar_game_over},
        {ARRIBA, jugando, moverAdelante},
        {ABAJO, jugando, moverAtras},
        {IZDA, jugando, moverIzda},
        {DCHA, jugando, moverDcha},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, jugando, do_nothing}}
```

Definition at line 269 of file fsm.c.

4.3.5.7 menu_ppal_esperando_opcion STATE menu_ppal_esperando_opcion

Initial value:

```
=
    {
        {CTE_OPCION, poniendo_nombre, ir_a_poniendo_nombre},
        {CTE_OPCION + 1, seleccionando_dificultad, ir_a_seleccionando_dificultad},
        {CTE_OPCION + 2, viendo_ranking, ir_a_viendo_ranking},
        {CTE_OPCION + 3, viendo_creditos, ir_a_viendo_creditos},
        {CTE_OPCION + 4, NULL, salir_del_juego},
        {FIN_TABLA, menu_ppal_esperando_opcion, do_nothing}
}
```

Definition at line 227 of file fsm.c.

4.3.5.8 poniendo_nombre STATE poniendo_nombre

Initial value:

```
=
    {
        {ESC, en_menu_ppal, ir_a_menu_ppal},
        {ENTER, jugando, iniciar_juego},
        {ARRIBA, poniendo_nombre, subirLetra},
        {ABAJO, poniendo_nombre, bajarLetra},
        {DCHA, poniendo_nombre, siguienteLetra},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, poniendo_nombre, agregarLetra}
}
```

Definition at line 258 of file fsm.c.

4.3.5.9 seleccionando_dificultad STATE seleccionando_dificultad

Initial value:

```
=
    {
        {ENTER, en_menu_ppal, procesar_enter_dificultad},
        {ARRIBA, seleccionando_dificultad, subirOpcion},
        {ABAJO, seleccionando_dificultad, bajarOpcion},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, seleccionando_dificultad, do_nothing}}
```

Definition at line 238 of file fsm.c.

4.3.5.10 viendo_creditos STATE viendo_creditos

Initial value:

```
=
    {
        {ENTER, en_menu_ppal, procesar_enter_creditos},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, viendo_creditos, do_nothing}}
```

Definition at line 252 of file fsm.c.

4.3.5.11 viendo_ranking STATE viendo_ranking

Initial value:

```
=
    {
        {ENTER, en_menu_ppal, procesar_enter_ranking},
        {FORCE_SALIR, NULL, salir_del_juego},
        {FIN_TABLA, viendo_ranking, do_nothing}}
```

Definition at line 246 of file fsm.c.

4.4 fsm.c

Go to the documentation of this file.

```
00001
00012 /*****************************************************************************
00013  * INCLUDE HEADER FILES
00014  *****************************************************************************/
00015
00016 #include "fsm.h"
00017
00018 #include "display.h"
00019 #include "game.h"
00020 #include "menu.h"
00021 #include "input.h"
00022 #include "nombre.h"
00023 #include "sound.h"
00024 #include "ranking.h"
00025
00026 #include <stdio.h>
00027 #include <stdlib.h>
00028 #include <stdint.h>
00029 #include <string.h>
00030 #include <pthread.h>
00031 #include <unistd.h>
00032
00033 /*****************************************************************************
00034  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00035  *****************************************************************************/
00036
00037 // Codigo para indicar que se llego al final de la tabla de estados
```

```
00038 #define FIN_TABLA 0xFF
00039
00040 // Para offsetear estados relativos al menu
00041 #define CTE_OPCION 100
00042
00043 // Delay en us que fixea consumo de CPU
00044 #define FIX_CPU_USAGE_SLEEP_US 500
00045
00046 /*****************************************************************************
00047  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00048  *****************************************************************************/
00049
00050 typedef struct state_diagram_edge STATE;
00051
00052 // Estructura genérica de un estado de la FSM.
00053 struct state_diagram_edge
00054 {
00055     event_t evento;
00056     STATE *proximo_estado;
00057     void (*p_rut_accion)(void);
00058 };
00059
00060 #pragma region privatePrototypes
00061 /*****************************************************************************
00062  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00063  *****************************************************************************/
00064
00071 static void *threadInput(void *ptr);
00072
00079 static void *threadJuego(void *ptr);
00080
00086 static void *threadDisplayRanking(void *ptr);
00087
00093 static void *threadDisplayCreditos(void *ptr);
00094
00100 static void do_nothing(void);
00101
00106 static void procesar_enter_menu(void);
00107
00112 static void ir_a_menu_ppal(void);
00113
00118 static void ir_a_poniendo_nombre(void);
00119
00124 static void ir_a_seleccionando_dificultad(void);
00125
00130 static void ir_a_viendo_ranking(void);
00131
00136 static void ir_a_viendo_creditos(void);
00137
00142 static void salir_del_juego(void);
00143
00148 static void procesar_enter_dificultad(void);
00149
00154 static void procesar_enter_ranking(void);
00155
00160 static void procesar_enter_creditos(void);
00161
00166 static void iniciar_juego(void);
00167
00172 static void pausar(void);
00173
00179 static void continuar(void);
00180
00185 static void procesar_game_over(void);
00186
00187 #pragma endregion privatePrototypes
00188
00189 /*****************************************************************************
00190  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00191  *****************************************************************************/
00192
00193 // Puntero al estado actual
00194 static STATE *p2CurrentState = NULL;
00195
00196 // Threads implementados
00197 static pthread_t tinput, tjuego, tdisplayranking, tdisplaycreditos;
00198
00199 #pragma region FSM STATES
00200 /*****************************************************************************
00201  * FSM STATES
00202  *****************************************************************************/
00203
00204 // Forward declarations de los estados
00205 extern STATE en_menu_ppal[];
00206 extern STATE menu_ppal_esperando_opcion[];
00207 extern STATE seleccionando_dificultad[];
00208 extern STATE viendo_ranking[];
```

```
00209 extern STATE viendo_creditos[];
00210 extern STATE poniendo_nombre[];
00211
00212 extern STATE jugando[];
00213 extern STATE en_pausa[];
00214 extern STATE en_pausa_esperando_opcion[];
00215 extern STATE en_game_over[];
00216 extern STATE en_game_over_esperando_opcion[];
00217 // Forward declarations de los estados
00218
00219 STATE en_menu_ppal[] =
00220     {
00221         {ENTER, menu_ppal_esperando_opcion, procesar_enter_menu},
00222         {ARRIBA, en_menu_ppal, subirOpcion},
00223         {ABAJO, en_menu_ppal, bajarOpcion},
00224         {FORCE_SALIR, NULL, salir_del_juego},
00225         {FIN_TABLA, en_menu_ppal, do_nothing}};
00226
00227 STATE menu_ppal_esperando_opcion[] =
00228     {
00229         {CTE_OPCION, poniendo_nombre, ir_a_poniendo_nombre},
00230         {CTE_OPCION + 1, seleccionando_dificultad, ir_a_seleccionando_dificultad},
00231         {CTE_OPCION + 2, viendo_ranking, ir_a_viendo_ranking},
00232         {CTE_OPCION + 3, viendo_creditos, ir_a_viendo_creditos},
00233         {CTE_OPCION + 4, NULL, salir_del_juego},
00234         {FIN_TABLA, menu_ppal_esperando_opcion, do_nothing}
00235
00236 };
00237
00238 STATE seleccionando_dificultad[] =
00239     {
00240         {ENTER, en_menu_ppal, procesar_enter_dificultad},
00241         {ARRIBA, seleccionando_dificultad, subirOpcion},
00242         {ABAJO, seleccionando_dificultad, bajarOpcion},
00243         {FORCE_SALIR, NULL, salir_del_juego},
00244         {FIN_TABLA, seleccionando_dificultad, do_nothing}};
00245
00246 STATE viendo_ranking[] =
00247     {
00248         {ENTER, en_menu_ppal, procesar_enter_ranking},
00249         {FORCE_SALIR, NULL, salir_del_juego},
00250         {FIN_TABLA, viendo_ranking, do_nothing}};
00251
00252 STATE viendo_creditos[] =
00253     {
00254         {ENTER, en_menu_ppal, procesar_enter_creditos},
00255         {FORCE_SALIR, NULL, salir_del_juego},
00256         {FIN_TABLA, viendo_creditos, do_nothing}};
00257
00258 STATE poniendo_nombre[] =
00259     {
00260         {ESC, en_menu_ppal, ir_a_menu_ppal},
00261         {ENTER, jugando, iniciar_juego},
00262         {ARRIBA, poniendo_nombre, subirLetra},
00263         {ABAJO, poniendo_nombre, bajarLetra},
00264         {DCHA, poniendo_nombre, siguienteLetra},
00265         {FORCE_SALIR, NULL, salir_del_juego},
00266         {FIN_TABLA, poniendo_nombre, agregarLetra} // Si no coincide el evento con ninguna de las
      teclas previas, se toam como si se apretase una letra
00267 };
00268
00269 STATE jugando[] =
00270     {
00271         {ENTER, en_pausa, pausar},
00272         {GAME_OVER, en_game_over, procesar_game_over},
00273         {ARRIBA, jugando, moverAdelante},
00274         {ABAJO, jugando, moverAtras},
00275         {IZDA, jugando, moverIzda},
00276         {DCHA, jugando, moverDcha},
00277         {FORCE_SALIR, NULL, salir_del_juego},
00278         {FIN_TABLA, jugando, do_nothing}};
00279
00280 STATE en_pausa[] =
00281     {
00282         {ENTER, en_pausa_esperando_opcion, procesar_enter_menu},
00283         {ARRIBA, en_pausa, subirOpcion},
00284         {ABAJO, en_pausa, bajarOpcion},
00285         {FORCE_SALIR, NULL, salir_del_juego},
00286         {FIN_TABLA, en_pausa, do_nothing}};
00287
00288 STATE en_pausa_esperando_opcion[] =
00289     {
00290         {CTE_OPCION, jugando, continuar},
00291         {CTE_OPCION + 1, jugando, iniciar_juego},
00292         {CTE_OPCION + 2, en_menu_ppal, ir_a_menu_ppal},
00293         {FIN_TABLA, en_pausa_esperando_opcion, do_nothing}};
00294
```

```
00295 STATE en_game_over[] =
00296     {
00297         {ENTER, en_game_over_esperando_opcion, procesar_enter_menu},
00298         {ARRIBA, en_game_over, subirOpcion},
00299         {ABAJO, en_game_over, bajarOpcion},
00300         {FORCE_SALIR, NULL, salir_del_juego},
00301         {FIN_TABLA, en_game_over, do_nothing}};
00302
00303 STATE en_game_over_esperando_opcion[] =
00304     {
00305         {CTE_OPCION, jugando, iniciar_juego},
00306         {CTE_OPCION + 1, en_menu_ppal, ir_a_menu_ppal},
00307         {FIN_TABLA, en_game_over_esperando_opcion, do_nothing}};
00308 #pragma endregion FSM STATES
00309
00310 /*******************************************************************************
00311  *******************************************************************************
00312                         GLOBAL FUNCTION DEFINITIONS
00313  *******************************************************************************
00314  *******************************************************************************/
00315
00316 bool inicializarFsm(void)
00317 {
00318     p2CurrentState = en_menu_ppal;
00319
00320     srand(time(NULL));
00321
00322     iniciarDisplay();
00323     iniciarMenu();
00324     iniciarEntradas();
00325     iniciarSonido();
00326
00327     iniciarRanking();
00328
00329     ir_a_menu_ppal();
00330
00331     pthread_create(&tinput, NULL, threadInput, NULL);
00332
00333     return true;
00334 }
00335
00336 void fsm(event_t evento_actual)
00337 {
00338     STATE *aux = p2CurrentState;
00339     /*
00340     Mientras el evento actual no coincida con uno "interesante", y mientras no se haya recorrido
00341     todo el estado...
00342     */
00343     while ((aux->evento != evento_actual) && (aux->evento != FIN_TABLA))
00344         // Verifico con la siguiente posibilidad dentro del mismo estado.
00345         ++aux;
00346
00347     // Pasa al siguiente estado
00348     p2CurrentState = aux->proximo_estado;
00349
00350     // Ejecuta la rutina correspondiente
00351     (*aux->p_rut_accion)();
00352 }
00353
00354 void fixHighCpuUsage(void)
00355 {
00356     usleep(FIX_CPU_USAGE_SLEEP_US);
00357 }
00358
00359 /*******************************************************************************
00360  *******************************************************************************
00361                         LOCAL FUNCTION DEFINITIONS
00362  *******************************************************************************
00363  *******************************************************************************/
00364
00365 static void *threadInput(void *ptr)
00366 {
00367     while (p2CurrentState)
00368     {
00369         event_t entrada = leerEntradas();
00370         if (entrada != NADA)
00371             queueInsertar(entrada);
00372
00373         fixHighCpuUsage();
00374     }
00375
00376     return NULL;
00377 }
00378
00379 static void *threadJuego(void *ptr)
00380 {
00381
```

```
00382     reconfigurarDisplayON();
00383
00384     srand(time(NULL));
00385
00386     while (p2CurrentState == jugando)
00387     {
00388         if (tiempoRefrescoEntidades())
00389             refrescar();
00390
00391         actualizarInterfaz();
00392
00393         fixHighCpuUsage();
00394     }
00395
00396     pausarJuego();
00397
00398     reconfigurarDisplayOFF();
00399
00400     return NULL;
00401 }
00402
00403 static void *threadDisplayRanking(void *ptr)
00404 {
00405     reconfigurarDisplayON();
00406
00407     cargarRanking();
00408
00409     while (p2CurrentState == viendo_ranking)
00410         mostrarRanking();
00411
00412     limpiarDisplay();
00413
00414     reconfigurarDisplayOFF();
00415
00416     return NULL;
00417 }
00418
00419 static void *threadDisplayCreditos(void *ptr)
00420 {
00421     reconfigurarDisplayON();
00422
00423     cargarCreditos();
00424
00425     while (p2CurrentState == viendo_creditos)
00426     {
00427         mostrarCreditos();
00428         fixHighCpuUsage();
00429     }
00430
00431     reconfigurarDisplayOFF();
00432
00433     return NULL;
00434 }
00435
00436 static void do_nothing(void)
00437 {
00438 }
00439
00440 static void procesar_enter_menu(void)
00441 {
00442     reproducirEfecto(EFECTO_MENU_ENTER);
00443     queueInsertar(CTE_OPCION + getOpcion());
00444 }
00445
00446 static void ir_a_menu_ppal()
00447 {
00448     limpiarDisplay();
00449     dejarTexto("MENU", POS_MSJ_MENU, true);
00450     reproducirMusica(MUSICA_MENU_PPAL);
00451     int menu[5] = {JUGAR, DIFICULTAD, RANKING, CREDITOS, SALIRTXT};
00452     setMenu(menu, 5);
00453     setOpcion(0);
00454 }
00455
00456 static void ir_a_viendo_ranking()
00457 {
00458     limpiarDisplay();
00459     reconfigurarDisplayOFF();
00460     reproducirMusica(MUSICA_RANKING);
00461     pthread_create(&tdisplayranking, NULL, threadDisplayRanking, NULL);
00462 }
00463
00464 static void ir_a_viendo_creditos(void)
00465 {
00466     limpiarDisplay();
00467     reconfigurarDisplayOFF();
00468     reproducirMusica(MUSICA_CREDITOS);
```

```
00469     pthread_create(&tdisplaycreditos, NULL, threadDisplayCreditos, NULL);
00470 }
00471
00472 static void salir_del_juego()
00473 {
00474     pthread_join(tinput, NULL);
00475     pausarMusica();
00476     //reproducirEfecto(EFECTO_SALIENDO);
00477     //sleep(2);
00478     destruirMenu();
00479     destruirSonido();
00480     desiniciarRanking();
00481     limpiarDisplay();
00482     queueInsertar(SALIR);
00483 }
00484
00485 static void procesar_enter_ranking(void)
00486 {
00487     pthread_join(tdisplayranking, NULL);
00488     reconfigurarDisplayON();
00489     ir_a_menu_ppal();
00490 }
00491
00492 static void procesar_enter_creditos(void)
00493 {
00494     pthread_join(tdisplaycreditos, NULL);
00495     reconfigurarDisplayON();
00496     ir_a_menu_ppal();
00497 }
00498
00499 static void iniciar_juego(void)
00500 {
00501     limpiarDisplay();
00502     char *nombreJugador = devolverNombre();
00503     if (nombreJugador == NULL)
00504         setNombre(DEFAULT_PLAYER_NAME);
00505     else if (nombreJugador[0] == 0)
00506         setNombre(DEFAULT_PLAYER_NAME);
00507     else
00508         setNombre(nombreJugador);
00509
00510     if (verificarJugadorRanking(getNombre()))
00511         setMaxPuntos(getJugadorRankingPuntos(getNombre()));
00512
00513     inicializarJuego();
00514     reconfigurarDisplayOFF();
00515
00516     reproducirMusica(MUSICA_JUGANDO);
00517
00518     reiniciarNivel();
00519     pthread_create(&tjuego, NULL, threadJuego, NULL);
00520 }
00521
00522 static void ir_a_poniendo_nombre()
00523 {
00524     limpiarDisplay();
00525     dejarTexto("INGRESE NOMBRE", POS_MSJ_NOMBRE, true);
00526     nuevoNombre();
00527 }
00528
00529 static void ir_a_seleccionando_dificultad()
00530 {
00531     limpiarDisplay();
00532     dejarTexto("DIFICULTAD", POS_MSJ_DIFICULTAD, true);
00533     int menu[3] = {FACIL, NORMAL, DIFICIL};
00534     setMenu(menu, 3);
00535     setOpcion(0);
00536 }
00537
00538 static void procesar_enter_dificultad(void)
00539 {
00540     setDificultad(getOpcion());
00541     reproducirEfecto(EFECTO_MENU_ENTER);
00542     ir_a_menu_ppal();
00543 }
00544
00545 static void pausar(void)
00546 {
00547     limpiarDisplay();
00548     pthread_join(tjuego, NULL);
00549     reproducirMusica(MUSICA_MENU_PAUSA);
00550     reconfigurarDisplayON();
00551     dejarTexto("PAUSA", POS_MSJ_PAUSA, true);
00552     int menu[3] = {CONTINUAR, REINICIAR, SALIRTXT};
00553     setMenu(menu, 3);
00554     setOpcion(0);
00555 }
```

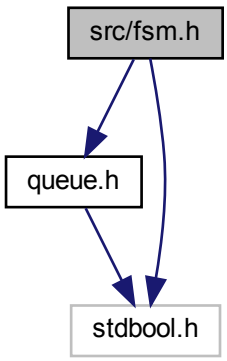
```
00556
00557 static void continuar(void)
00558 {
00559     limpiarDisplay();
00560     reconfigurarDisplayOFF();
00561     reproducirMusica(MUSICA_JUGANDO);
00562     reanudarJuego();
00563     pthread_create(&tjuego, NULL, threadJuego, NULL);
00564 }
00565
00566 static void procesar_game_over(void)
00567 {
00568     pthread_join(tjuego, NULL);
00569
00570     limpiarDisplay();
00571
00572     reproducirMusica(MUSICA_GAME_OVER);
00573     reconfigurarDisplayON();
00574
00575     unsigned long long jugador_puntos = getPuntos();
00576
00577     if (jugador_puntos > getMaxPuntos())
00578     {
00579         reproducirEfecto(EFECTO_NUEVO_MAX_SCORE);
00580
00581         mostrarTexto("NUEVA PUNTUACION ALTA", POS_MSJ_NEW_HI_SCORE);
00582         setMaxPuntos(jugador_puntos);
00583
00584         actualizarRanking(getNombre(), getMaxPuntos());
00585     }
00586
00587     mostrarTexto("FIN DEL JUEGO", POS_MSJ_GAME_OVER);
00588     int menu[2] = {REINICIAR, SALIRTXT};
00589     limpiarDisplay();
00590     setMenu(menu, 2);
00591     setOpcion(0);
00592 }
```

4.5 src/fsm.h File Reference

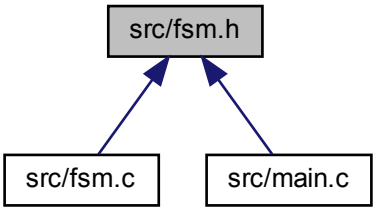
Header del modulo fsm. Contiene los prototipos de funciones necesarias para iniciar la fsm desde main.c.

```
#include "queue.h"
#include <stdbool.h>
```

Include dependency graph for fsm.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool inicializarFsm (void)

  Inicializa la FSM, notificando si tuvo exito.
- void fsm (event_t evento_actual)

  Intérprete de la FSM. Se encarga de hacer correr la FSM a partir de un estado y evento dados.
- void fixHighCpuUsage (void)

  Fixea consumo elevado de cpu en el while loop principal.

4.5.1 Detailed Description

Header del modulo fsm. Contiene los prototipos de funciones necesarias para iniciar la fsm desde main.c.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file fsm.h.

4.5.2 Function Documentation

4.5.2.1 fixHighCpuUsage() `void fixHighCpuUsage ( void )`

Fixea consumo elevado de cpu en el while loop principal.

Not the best solucion, but sirve...

https://softwareengineering.stackexchange.com/questions/256524/infinite-while-loop-cpu-

Definition at line 354 of file fsm.c.

4.5.2.2 fsm()

```
void fsm (
        event_t evento_actual )
```

Intérprete de la FSM. Se encarga de hacer correr la FSM a partir de un estado y evento dados.

Parameters

p_tabla_estado	Estado actual.
evento_actual	Evento recibido.

Definition at line 336 of file fsm.c.

4.5.2.3 inicializarFsm() `bool inicializarFsm ( void )`

Inicializa la FSM, notificando si tuvo exito.

Returns

true Exito

false Error

Definition at line 316 of file fsm.c.

4.6 fsm.h

Go to the documentation of this file.

```
00001
00013 #ifndef _FSM_H_
00014 #define _FSM_H_
00015
00016 /*******************************************************************************
00017  * INCLUDE HEADER FILES
00018  ******************************************************************************/
00019
00020 #include "queue.h"
00021 #include <stdbool.h>
00022
00023 /*******************************************************************************
00024  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00025  ******************************************************************************/
00026
00033 bool inicializarFsm(void);
00034
00041 void fsm(event_t evento_actual);
00042
00051 void fixHighCpuUsage(void);
00052
00053 /*******************************************************************************
00054  ******************************************************************************/
00055
00056 #endif // _FSM_H_
```

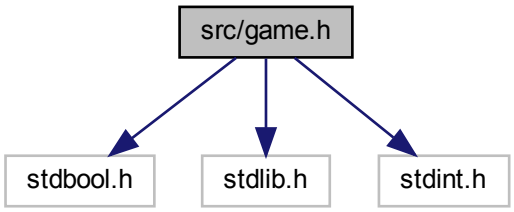
4.7 src/game.h File Reference

Header del modulo game Vinculo entre la fsm y las plataformas en lo que respecta a la informacion del jugador y el progreso del juego.

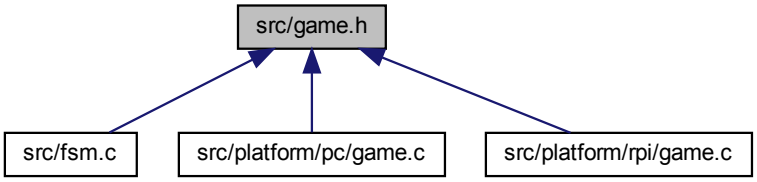
```
#include <stdbool.h>
#include <stdlib.h>
```

```
#include <stdint.h>
```

Include dependency graph for game.h:



This graph shows which files directly or indirectly include this file:



Functions

- void setNombre (char *nombre)

  Confirma el nombre del jugador.
- void setMaxPuntos (unsigned long long max)

  Setea los puntos maximos del jugador.
- void setDificultad (int dif)

  Setea la dificultad a usar.
- bool tiempoRefrescoEntidades (void)

  Chequea si es tiempo de refrescar entidades según la plataforma.
- char * getNombre (void)

  Devuelve el nombre del jugador.
- unsigned long long getPuntos (void)

  Devuelve el puntaje del jugador.
- unsigned long long getMaxPuntos (void)

  Devuelve el puntaje máximo del jugador.
- int getNivel (void)

  Devuelve el nivel//run del jugador.
- void inicializarJuego (void)

  Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.

- void reiniciarNivel (void)
  Configuraciones para reiniciar el nivel.
- void pausarJuego (void)
  Pausa el juego.
- void reanudarJuego (void)
  Saca el juego de pausa.
- void refrescar (void)
  Actualizaciones relativas a actualizar las entidades.
- void moverAdelante (void)
  Avanza el jugador.
- void moverAtras (void)
  Retrocede el jugador.
- void moverIzda (void)
  Mueve el jugador a la izquierda.
- void moverDcha (void)
  Mueve el jugador a la derecha.
- void respawn (void)
  Respawnea el jugador.
- void perderVida (void)
  Resta una vida.
- void actualizarInterfaz (void)
  Actualizaciones relativas a lo visual.

4.7.1 Detailed Description

Header del modulo game Vinculo entre la fsm y las plataformas en lo que respecta a la informacion del jugador y el progreso del juego.

Header del modulo genérico menu. Prototipos de funciones de interaccion con el menu del juego.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file game.h.

4.7.2 Function Documentation

4.7.2.1 actualizarInterfaz() `void actualizarInterfaz ( void )`

Actualizaciones relativas a lo visual.

Definition at line 155 of file game.c.

4.7.2.2 getMaxPuntos() `unsigned long long getMaxPuntos ( void )`

Devuelve el puntaje máximo del jugador.

Returns

unsigned long long

Definition at line 82 of file game.c.

4.7.2.3 getNivel() `int getNivel ( void )`

Devuelve el nivel//run del jugador.

Returns

int

Definition at line 87 of file game.c.

4.7.2.4 getNombre() `char * getNombre ( void )`

Devuelve el nombre del jugador.

Returns

char*

Definition at line 72 of file game.c.

4.7.2.5 getPuntos() `unsigned long long getPuntos ( void )`

Devuelve el puntaje del jugador.

Returns

unsigned long long

Definition at line 77 of file game.c.

4.7.2.6 inicializarJuego() void inicializarJuego (void)

Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.

Definition at line 92 of file game.c.

4.7.2.7 moverAdelante() void moverAdelante (void)

Avanza el jugador.

Definition at line 131 of file game.c.

4.7.2.8 moverAtras() void moverAtras (void)

Retrocede el jugador.

Definition at line 136 of file game.c.

4.7.2.9 moverDcha() void moverDcha (void)

Mueve el jugador a la derecha.

Definition at line 146 of file game.c.

4.7.2.10 moverIzda() void moverIzda (void)

Mueve el jugador a la izquierda.

Definition at line 141 of file game.c.

4.7.2.11 pausarJuego() void pausarJuego (void)

Pausa el juego.

Definition at line 101 of file game.c.

4.7.2.12 perderVida() `void perderVida (`

`void )`

Resta una vida.

Definition at line 303 of file game.c.

4.7.2.13 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 188 of file game.c.

4.7.2.14 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 114 of file game.c.

4.7.2.15 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 105 of file game.c.

4.7.2.16 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 151 of file game.c.

4.7.2.17 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 47 of file game.c.

4.7.2.18 setMaxPuntos() `void setMaxPuntos ( unsigned long long max )`

Setea los puntos maximos del jugador.

Parameters

max	

Definition at line 42 of file game.c.

4.7.2.19 setNombre() `void setNombre ( char * nombre )`

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 37 of file game.c.

4.7.2.20 tiempoRefrescoEntidades() `bool tiempoRefrescoEntidades ( void )`

Chequea si es tiempo de refrescar entidades según la plataforma.

Returns

true

false

Definition at line 67 of file game.c.

4.8 game.h

Go to the documentation of this file.

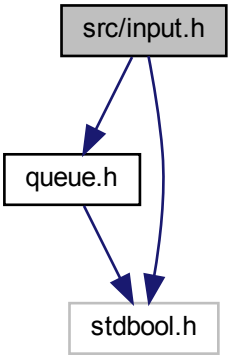
```
00001
00013 #ifndef _GAME_H_
00014 #define _GAME_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdbool.h>
00021 #include <stdlib.h>
00022 #include <stdint.h>
00023
00024 /*****************************************************************************
00025  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00026  *****************************************************************************/
00027
00033 void setNombre(char *nombre);
00034
00040 void setMaxPuntos(unsigned long long max);
00041
00047 void setDificultad(int dif);
00048
00055 bool tiempoRefrescoEntidades(void);
00056
00062 char *getNombre(void);
00063
00069 unsigned long long getPuntos(void);
00070
00076 unsigned long long getMaxPuntos(void);
00077
00083 int getNivel(void);
00084
00089 void inicializarJuego(void);
00090
00095 void reiniciarNivel(void);
00096
00101 void pausarJuego(void);
00102
00107 void reanudarJuego(void);
00108
00113 void refrescar(void);
00114
00119 void moverAdelante(void);
00120
00125 void moverAtras(void);
00126
00131 void moverIzda(void);
00132
00137 void moverDcha(void);
00138
00143 void respawn(void);
00144
00149 void perderVida(void);
00150
00155 void actualizarInterfaz(void);
00156
00157 /*****************************************************************************
00158  *****************************************************************************/
00159
00160 #endif // _GAME_H_
```

4.9 src/input.h File Reference

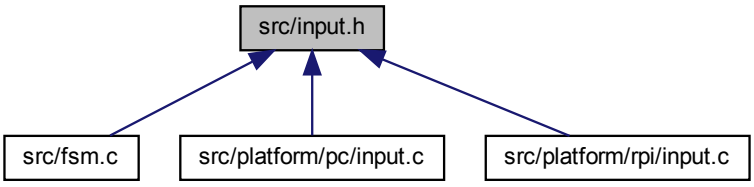
Header del modulo input Vinculo entre la fsm y las plataformas en lo que respecta al manejo de acciones externas.

```
#include "queue.h"
#include <stdbool.h>
```

Include dependency graph for input.h:



This graph shows which files directly or indirectly include this file:



Functions

- void iniciarEntradas (void)

  Inicializa las entradas de la plataforma.
- event_t leerEntradas (void)

  Devuelve una entrada válida.

4.9.1 Detailed Description

Header del modulo input Vinculo entre la fsm y las plataformas en lo que respecta al manejo de acciones externas.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file input.h.

4.9.2 Function Documentation

4.9.2.1 iniciarEntradas() `void iniciarEntradas ( void )`

Inicializa las entradas de la plataforma.

Definition at line 37 of file input.c.

4.9.2.2 leerEntradas() `event_t leerEntradas ( void )`

Devuelve una entrada válida.

Returns

event_t enum eventos_tecla

Definition at line 41 of file input.c.

4.10 input.h

Go to the documentation of this file.

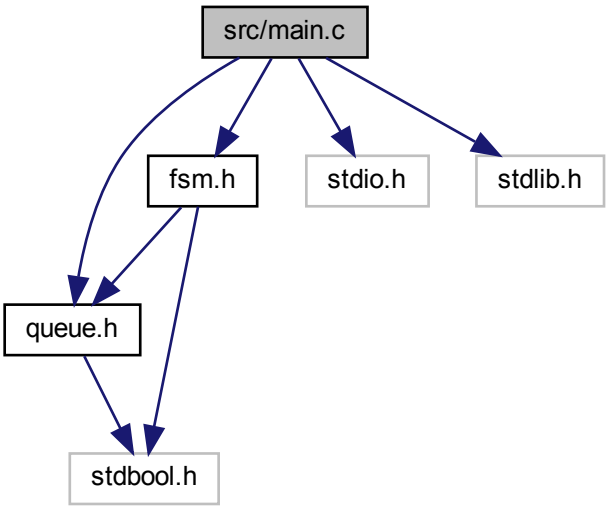
```
00001
00013 #ifndef _INPUT_H_
00014 #define _INPUT_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include "queue.h"
00021
00022 #include <stdbool.h>
00023
00024 /*****************************************************************************
00025  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00026  *****************************************************************************/
00027
00033 void iniciarEntradas(void);
00034
00040 event_t leerEntradas(void);
00041
00042 /*****************************************************************************
00043  *****************************************************************************/
00044
00045 #endif // _INPUT_H_
```

4.11 src/main.c File Reference

Archivo principal. Inicia y pone a correr la maquian de estados (fsm).

```
#include "fsm.h"
#include "queue.h"
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for main.c:



Functions

- int main (void)

4.11.1 Detailed Description

Archivo principal. Inicia y pone a correr la maquian de estados (fsm).

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file main.c.

4.11.2 Function Documentation

4.11.2.1 main() int main (void)

Definition at line 28 of file main.c.

4.12 main.c

Go to the documentation of this file.

```
00001
00012 /*****************************************************************************
00013  * INCLUDE HEADER FILES
00014  *****************************************************************************/
00015
00016 #include "fsm.h"
00017 #include "queue.h"
00018
00019 #include <stdio.h>
00020 #include <stdlib.h>
00021
00022 /*****************************************************************************
00023  *****************************************************************************
00024                                     MAIN
00025  *****************************************************************************
00026  *****************************************************************************/
00027
00028 int main(void)
00029 {
00030     event_t evento;
00031
00032     if (!inicializarFsm())
00033         return 1;
00034
00035     while ((evento = queueSiguienteEvento()))
00036     {
00037         if (evento != NADA)
00038         {
00039             fsm(evento);
00040         }
00041
00042         fixHighCpuUsage();
00043     }
00044
00045     destruirQueue();
00046
00047     printf("\nGracias por jugar <3\n");
00048
00049     return 0;
00050 }
```

4.13 menu.h

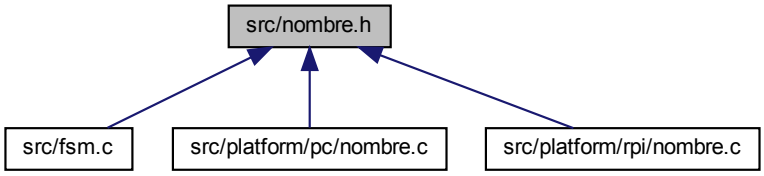
```
00001
00012 #ifndef _MENU_H_
00013 #define _MENU_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include "queue.h"
00020
00021 /*****************************************************************************
00022  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00023  *****************************************************************************/
00024
00025 enum textos_menu
00026 {
00027     JUGAR = 0,
00028     DIFICULTAD,
```

```
00029     RANKING,
00030     CREDITOS,
00031     SALIRTXT,
00032     CONTINUAR,
00033     REINICIAR,
00034     FACIL,
00035     NORMAL,
00036     DIFICIL
00037 };
00038
00039 /*******************************************************************************
00040  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00041  ******************************************************************************/
00042
00047 void iniciarMenu(void);
00048
00053 void destruirMenu(void);
00054
00061 void setMenu(int *a, unsigned int size);
00062
00068 void setOpcion(int opc);
00069
00075 int getOpcion(void);
00076
00081 void subirOpcion(void);
00082
00087 void bajarOpcion(void);
00088
00089 /*******************************************************************************
00090  ******************************************************************************/
00091
00092 #endif // _MENU_H_
```

4.14 src/nombre.h File Reference

Header del modulo genérico nombre. Prototipos de funciones de relacionadas al ingreso del nombre del jugador.

This graph shows which files directly or indirectly include this file:



Functions

- void nuevoNombre (void)

  Se ejecuta una vez al ingresar a poner un nuevo nombre.
- void subirLetra (void)

  Selecciona la siguiente letra superior.
- void bajarLetra (void)

  Selecciona la letra inferior.
- void siguienteLetra (void)

  Confirma la letra y pasa a seleccionar la siguiente.
- void agregarLetra (void)

  Confirma la letra.
- void subirNombre (void)
- char * devolverNombre (void)

  Devuelve puntero al nombre.

4.14.1 Detailed Description

Header del modulo genérico nombre. Prototipos de funciones de relacionadas al ingreso del nombre del jugador.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file nombre.h.

4.14.2 Function Documentation

4.14.2.1 agregarLetra() `void agregarLetra ( void )`

Confirma la letra.

Definition at line 61 of file nombre.c.

4.14.2.2 bajarLetra() `void bajarLetra ( void )`

Selecciona la letra inferior.

Definition at line 53 of file nombre.c.

4.14.2.3 devolverNombre() `char * devolverNombre ( void )`

Devuelve puntero al nombre.

Returns

char∗ Puntero al nombre

Definition at line 82 of file nombre.c.

4.14.2.4 nuevoNombre() `void nuevoNombre (`

`void )`

Se ejecuta una vez al ingresar a poner un nuevo nombre.

Definition at line 36 of file nombre.c.

4.14.2.5 siguienteLetra() `void siguienteLetra (`

`void )`

Confirma la letra y pasa a seleccionar la siguiente.

Definition at line 57 of file nombre.c.

4.14.2.6 subirLetra() `void subirLetra (`

`void )`

Selecciona la siguiente letra superior.

Definition at line 49 of file nombre.c.

4.14.2.7 subirNombre() `void subirNombre (`

`void )`

Definition at line 78 of file nombre.c.

4.15 nombre.h

Go to the documentation of this file.

```
00001
00013 #ifndef _NOMBRE_H_
00014 #define _NOMBRE_H_
00015
00016 /*****************************************************************************
00017  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00018  *****************************************************************************/
00019
00024 void nuevoNombre(void);
00025
00030 void subirLetra(void);
00031
00036 void bajarLetra(void);
00037
00042 void siguienteLetra(void);
00043
00048 void agregarLetra(void);
00049
00054 void subirNombre(void);
00055
00061 char *devolverNombre(void);
00062
00063 /*****************************************************************************
00064  *****************************************************************************/
00065
00066 #endif // _NOMBRE_H_
```

4.16 src/platform/pc/allegro_stuff.c File Reference

Source del modulo allegro_stuff. Se encarga de todas las inicializaciones y desinicializaciones relativas a allegro5 y sus addons.

```
#include "allegro_stuff.h"
#include "geometry.h"
#include <string.h>
#include "./algif5/algif.h"
```

Include dependency graph for allegro_stuff.c:

Data Structures

- struct allegro_t
- struct sounds_t

Macros

- #define FONT_HEIGHT 16
- #define SOUND_STREAM_FILE_CREDITS "credits_theme"
- #define SOUND_STREAM_FILE_MAIN "main_menu_theme"
- #define SOUND_STREAM_FILE_PAUSE "pause_menu_theme"
- #define SOUND_STREAM_FILE_PLAYING "playing_theme"
- #define SOUND_STREAM_FILE_RANKING "ranking_theme"
- #define SOUND_STREAM_FILE_RICK "rick"
- #define SOUND_STREAM_FILE_GAME_OVER "game_over"
- #define FONT_FILE_NAME "PublicPixel.ttf"
- #define SPRITE_HEART "minecraft_heart"
- #define SPRITE_BACKGROUND "sprite_background"
- #define SPRITE_CAR "sprite_cars"
- #define SPRITE_FROG "sprite_frog"
- #define SPRITE_LOG "sprite_log"
- #define SPRITE_TURTLES "sprite_turtles"
- #define SPRITE_MENU_HOME_BACK "sprite_menu_home_background"
- #define SPRITE_MENU_HOME "sprite_menu_home"
- #define SPRITE_MENU_DIFF_BACK "sprite_menu_diff_background"
- #define SPRITE_MENU_DIFF "sprite_menu_diff"
- #define SPRITE_MENU_PAUSE_BACK "sprite_menu_pause_background"
- #define SPRITE_MENU_PAUSE "sprite_menu_pause"
- #define SPRITE_MENU_GAME_OVER_BACK "sprite_menu_gameover_background"
- #define SPRITE_MENU_GAME_OVER "sprite_menu_gameover"
- #define SPRITE_CREDITS "sprite_credits"
- #define SPRITE_NAME "sprite_name"
- #define SPRITE_ICON "icon"
- #define SPRITE_DEAD "sprite_dead"

- #define SPRITE_BORDER "sprite_border"
- #define SPRITE_SPLASH "sprite_splash"
- #define SPRITE_COIN "sprite_coin"
- #define EXTENSION_SOUND_SAMPLE ".wav"
- #define EXTENSION_SOUND_STREAM ".opus"
- #define EXTENSION_SPRITES ".png"
- #define PATH_SOUND_STREAMS "../res/sounds/streams/"
- #define PATH_SOUND_SAMPLES "../res/sounds/samples/"
- #define PATH_FONTS "../res/fonts/"
- #define PATH_SPRITES "../res/sprites/"
- #define PATH_GIFS "../res/gifs/"
- #define GLOBAL_STREAM_VOLUME (double)0.5

Enumerations

- enum **SOUND_STREAM_STATES** { **SOUND_STREAM_STATE_NO_INIT** , **SOUND_STREAM_STATE_INIT** , **SOUND_STREAM_STATE_PAUSE** , **SOUND_STREAM_STATE_PLAY** }

Functions

- void must_init (bool test, const char *description)

  Inicializa "cosas", y sale del programa si hay problemas, avisando dónde estuvo.
- void allegro_inits (void)

  Inicializaciones de allegro. Si algo falla, lo notifica por consola y finaliza el programa.
- void allegro_deinits (void)

  Desinicializaciones de allegro.
- void allegro_reinit_display (void)

  Reinicializa el display de allegro.
- void allegro_deinit_display (void)

  Desinicializa el display.
- unsigned char allegro_get_last_key (void)

  Devuelve la ultima tecla presionada registrada.
- void allegro_set_last_key (unsigned char allegro_key_code)

  Setea una ultima tecla presionada.
- ALLEGRO_EVENT_TYPE allegro_wait_for_event (void)

  Espera a que ocurra un evento y lo devuelve.
- ALLEGRO_EVENT * allegro_get_next_event (void)

  Devuelve el proximo evento de la cola, si es que existe. De no haber, devuele false.
- ALLEGRO_EVENT allegro_get_var_event (void)

  Devuelve el evento de allegro.
- bool allegro_get_var_done (void)

  Devuelve flag de finalización del programa.
- bool allegro_get_var_redraw (void)

  Devuelve flag de renderización.
- void allegro_set_var_done (bool state)

  Setea flag de finalización del programa.
- void allegro_set_var_redraw (bool state)

  Setea flag de renderizacion.
- ALLEGRO_FONT * allegro_get_var_font (void)

  Devuelve la fuente de allegro.
- int allegro_get_var_font_h (void)

Devuelve el alto de un caracter de la funete usada.

- int allegro_get_var_font_w (void)

  Devuelve ancho de un caracter de la fuente usada.
- void allegro_clear_display (void)

  Pone negro el display.
- void allegro_draw_background (void)

  Dibuja la imagen de fondo.
- void allegro_draw_menu_background (int window)

  Dibuja imagen de fondo de un menu dado (sin highlight en ninguna opcion)
- bool allegro_is_event_queueVacia (void)

  Informa si al cola de eventos está vacía o no.
- ALLEGRO_EVENT_QUEUE * allegro_get_event_queue (void)

  Devuelve puntero a la cola de eventos.
- void allegro_set_var_event (ALLEGRO_EVENT event)

  Carga un evento de allegro.
- void allegro_sound_set_stream_credits (void)

  Selecciona musica de creditos. Comienza pausada.
- void allegro_sound_set_stream_main_menu (void)

  Selecciona musica de menu. Comienza pausada.
- void allegro_sound_set_stream_pause_menu (void)

  Selecciona musica de pausa. Comienza pausada.
- void allegro_sound_set_stream_ranking (void)

  Selecciona musica de ranking. Comienza pausada.
- void allegro_sound_set_stream_playing (void)

  Selecciona musica de jugando. Comienza pausada.
- void allegro_sound_set_stream_rick (void)
- void allegro_sound_set_stream_game_over (void)

  Selecciona musica de game over. Comienza pausada.
- void allegro_sound_toggle_stream (void)

  Cambia estado de reproduccion de la musica actual, si hay alguna seleccionada.
- void allegro_sound_play_stream (void)

  Reproduce la musica actual, si hay alguna seleccionada.
- void allegro_sound_pause_stream (void)

  Pausa la musica actual, si hay alguna seleccionada.
- void allegro_sound_restart_stream (void)

  Reinicia la musica actual, si hay alguna seleccionada.
- void allegro_sound_set_stream_gain_up (void)

  Aumenta en 0.1 la ganancia de stream.
- void allegro_sound_set_stream_gain_down (void)

  Reduce en 0.1 la ganancia de stream.
- void allegro_sound_play_effect_bonus (void)

  Reproduce efecto de bonus.
- void allegro_sound_play_effect_click (void)

  Reproduce efecto de click (seleccion de menu//aceptar//etc.)
- void allegro_sound_play_effect_crash (void)

  Reproduce efecto de choque.
- void allegro_sound_play_effect_drowned (void)

  Reproduce efecto de ahogado (toco agua)
- void allegro_sound_play_effect_goal (void)

  Reproduce efecto de 'llego a la meta'.
- void allegro_sound_play_effect_jump (void)

Reproduce efecto de salto.

- void allegro_sound_play_effect_low_time (void)

  Reproduce efecto de 'queda poco tiempo'.
- void allegro_sound_play_effect_run_completed (void)

  Reproduce efecto de 'run completada' (llego 5 veces a la meta)
- void allegro_sound_play_effect_menu_enter (void)

  Reproduce efecto de 'menu enter'.
- void allegro_sound_play_effect_new_max_score (void)

  Reproduce efecto de 'new_max_score'.
- void allegro_sound_play_effect_exiting (void)

  Reproduce efecto de 'saliendo'.
- void allegro_sound_play_effect_no_time (void)

  Reproduce efecto de 'sin tiempo'.
- void allegro_sound_play_effect_coin_drop (void)

  Reproduce efecto `de moneda tirada`
- void allegro_draw_hitbox (int x, int y, int w, int h)

  Dibuja un contorno rectangular.
- void allegro_rick_on (void)
- bool allegro_get_rick_flag (void)
- void allegro_set_rick_flag (bool state)
- void allegro_rick_off (void)
- void allegro_rick_draw (void)

Variables

- sprites_t sprites

4.16.1 Detailed Description

Source del modulo allegro_stuff. Se encarga de todas las inicializaciones y desinicializaciones relativas a allegro5 y sus addons.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file allegro_stuff.c.

4.16.2 Macro Definition Documentation

4.16.2.1 EXTENSION_SOUND_SAMPLE `#define EXTENSION_SOUND_SAMPLE ".wav"`

Definition at line 71 of file allegro_stuff.c.

4.16.2.2 EXTENSION_SOUND_STREAM `#define EXTENSION_SOUND_STREAM ".opus"`

Definition at line 72 of file allegro_stuff.c.

4.16.2.3 EXTENSION_SPRITES `#define EXTENSION_SPRITES ".png"`

Definition at line 73 of file allegro_stuff.c.

4.16.2.4 FONT_FILE_NAME `#define FONT_FILE_NAME "PublicPixel.ttf"`

Definition at line 39 of file allegro_stuff.c.

4.16.2.5 FONT_HEIGHT `#define FONT_HEIGHT 16`

Definition at line 28 of file allegro_stuff.c.

4.16.2.6 GLOBAL_STREAM_VOLUME `#define GLOBAL_STREAM_VOLUME (double)0.5`

Definition at line 82 of file allegro_stuff.c.

4.16.2.7 PATH_FONTS `#define PATH_FONTS "../res/fonts/"`

Definition at line 78 of file allegro_stuff.c.

4.16.2.8 PATH_GIFS `#define PATH_GIFS "../res/gifs/"`

Definition at line 80 of file allegro_stuff.c.

4.16.2.9 PATH_SOUND_SAMPLES `#define PATH_SOUND_SAMPLES "../res/sounds/samples/"`

Definition at line 77 of file allegro_stuff.c.

4.16.2.10 PATH_SOUND_STREAMS `#define PATH_SOUND_STREAMS "../res/sounds/streams/"`

Definition at line 76 of file allegro_stuff.c.

4.16.2.11 PATH_SPRITES `#define PATH_SPRITES "../res/sprites/"`

Definition at line 79 of file allegro_stuff.c.

4.16.2.12 SOUND_STREAM_FILE_CREDITS `#define SOUND_STREAM_FILE_CREDITS "credits_theme"`

Definition at line 31 of file allegro_stuff.c.

4.16.2.13 SOUND_STREAM_FILE_GAME_OVER `#define SOUND_STREAM_FILE_GAME_OVER "game_over"`

Definition at line 37 of file allegro_stuff.c.

4.16.2.14 SOUND_STREAM_FILE_MAIN `#define SOUND_STREAM_FILE_MAIN "main_menu_theme"`

Definition at line 32 of file allegro_stuff.c.

4.16.2.15 SOUND_STREAM_FILE_PAUSE `#define SOUND_STREAM_FILE_PAUSE "pause_menu_theme"`

Definition at line 33 of file allegro_stuff.c.

4.16.2.16 SOUND_STREAM_FILE_PLAYING `#define SOUND_STREAM_FILE_PLAYING "playing_theme"`

Definition at line 34 of file allegro_stuff.c.

4.16.2.17 SOUND_STREAM_FILE_RANKING `#define SOUND_STREAM_FILE_RANKING "ranking_theme"`

Definition at line 35 of file allegro_stuff.c.

4.16.2.18 SOUND_STREAM_FILE_RICK `#define SOUND_STREAM_FILE_RICK "rick"`

Definition at line 36 of file allegro_stuff.c.

4.16.2.19 SPRITE_BACKGROUND `#define SPRITE_BACKGROUND "sprite_background"`

Definition at line 43 of file allegro_stuff.c.

4.16.2.20 SPRITE_BORDER `#define SPRITE_BORDER "sprite_border"`

Definition at line 66 of file allegro_stuff.c.

4.16.2.21 SPRITE_CAR `#define SPRITE_CAR "sprite_cars"`

Definition at line 44 of file allegro_stuff.c.

4.16.2.22 SPRITE_COIN `#define SPRITE_COIN "sprite_coin"`

Definition at line 68 of file allegro_stuff.c.

4.16.2.23 SPRITE_CREDITS `#define SPRITE_CREDITS "sprite_credits"`

Definition at line 62 of file allegro_stuff.c.

4.16.2.24 SPRITE_DEAD `#define SPRITE_DEAD "sprite_dead"`

Definition at line 65 of file allegro_stuff.c.

4.16.2.25 SPRITE_FROG `#define SPRITE_FROG "sprite_frog"`

Definition at line 45 of file allegro_stuff.c.

4.16.2.26 SPRITE_HEART `#define SPRITE_HEART "minecraft_heart"`

Definition at line 42 of file allegro_stuff.c.

4.16.2.27 SPRITE_ICON `#define SPRITE_ICON "icon"`

Definition at line 64 of file allegro_stuff.c.

4.16.2.28 SPRITE_LOG `#define SPRITE_LOG "sprite_log"`

Definition at line 46 of file allegro_stuff.c.

4.16.2.29 SPRITE_MENU_DIFF `#define SPRITE_MENU_DIFF "sprite_menu_diff"`

Definition at line 57 of file allegro_stuff.c.

4.16.2.30 SPRITE_MENU_DIFF_BACK `#define SPRITE_MENU_DIFF_BACK "sprite_menu_diff_background"`

Definition at line 56 of file allegro_stuff.c.

4.16.2.31 SPRITE_MENU_GAME_OVER `#define SPRITE_MENU_GAME_OVER "sprite_menu_gameover"`

Definition at line 61 of file allegro_stuff.c.

4.16.2.32 SPRITE_MENU_GAME_OVER_BACK `#define SPRITE_MENU_GAME_OVER_BACK "sprite_menu_↩ gameover_background"`

Definition at line 60 of file allegro_stuff.c.

4.16.2.33 SPRITE_MENU_HOME `#define SPRITE_MENU_HOME "sprite_menu_home"`

Definition at line 49 of file allegro_stuff.c.

4.16.2.34 SPRITE_MENU_HOME_BACK `#define SPRITE_MENU_HOME_BACK "sprite_menu_home_background"`

Definition at line 48 of file allegro_stuff.c.

4.16.2.35 SPRITE_MENU_PAUSE `#define SPRITE_MENU_PAUSE "sprite_menu_pause"`

Definition at line 59 of file allegro_stuff.c.

4.16.2.36 SPRITE_MENU_PAUSE_BACK `#define SPRITE_MENU_PAUSE_BACK "sprite_menu_pause_↩ background"`

Definition at line 58 of file allegro_stuff.c.

4.16.2.37 SPRITE_NAME `#define SPRITE_NAME "sprite_name"`

Definition at line 63 of file allegro_stuff.c.

4.16.2.38 SPRITE_SPLASH `#define SPRITE_SPLASH "sprite_splash"`

Definition at line 67 of file allegro_stuff.c.

4.16.2.39 SPRITE_TURTLES `#define SPRITE_TURTLES "sprite_turtles"`

Definition at line 47 of file allegro_stuff.c.

4.16.3 Enumeration Type Documentation

4.16.3.1 SOUND_STREAM_STATES `enum SOUND_STREAM_STATES`

Definition at line 138 of file allegro_stuff.c.

4.16.4 Function Documentation

4.16.4.1 allegro_clear_display() `void allegro_clear_display (`

```
void )
```

Pone negro el display.

Definition at line 458 of file allegro_stuff.c.

4.16.4.2 allegro_deinit_display() `void allegro_deinit_display (`

```
void )
```

Desinicializa el display.

Definition at line 382 of file allegro_stuff.c.

4.16.4.3 allegro_deinits() `void allegro_deinits (`

```
void )
```

Desinicializaciones de allegro.

Definition at line 336 of file allegro_stuff.c.

4.16.4.4 allegro_draw_background() `void allegro_draw_background (`

```
void )
```

Dibuja la imagen de fondo.

Definition at line 463 of file allegro_stuff.c.

4.16.4.5 allegro_draw_hitbox() `void allegro_draw_hitbox (`

```
int x,
int y,
int w,
int h )
```

Dibuja un contorno rectangular.

Parameters

x	Topleft x
y	Topleft y
w	Ancho
h	Largo

Definition at line 738 of file allegro_stuff.c.

4.16.4.6 allegro_draw_menu_background() `void allegro_draw_menu_background ( int window )`

Dibuja imagen de fondo de un menu dado (sin highlight en ninguna opcion)

Parameters

window	enum MENU_WINDOWS

Definition at line 468 of file allegro_stuff.c.

4.16.4.7 allegro_get_event_queue() `ALLEGRO_EVENT_QUEUE * allegro_get_event_queue ( void )`

Devuelve puntero a la cola de eventos.

Definition at line 478 of file allegro_stuff.c.

4.16.4.8 allegro_get_last_key() `unsigned char allegro_get_last_key ( void )`

Devuelve la ultima tecla presionada registrada.

Returns

unsigned char ALLEGRO_KEY_CODE

Definition at line 391 of file allegro_stuff.c.

4.16.4.9 allegro_get_next_event() `ALLEGRO_EVENT * allegro_get_next_event ( void )`

Devuelve el proximo evento de la cola, si es que existe. De no haber, devuele false.

Returns

ALLEGRO_EVENT*

Definition at line 408 of file allegro_stuff.c.

4.16.4.10 allegro_get_rick_flag() `bool allegro_get_rick_flag ( void )`

Definition at line 756 of file allegro_stuff.c.

4.16.4.11 allegro_get_var_done() `bool allegro_get_var_done ( void )`

Devuelve flag de finalización del programa.

Returns

true Finaliza

false No finaliza

Definition at line 423 of file allegro_stuff.c.

4.16.4.12 allegro_get_var_event() `ALLEGRO_EVENT allegro_get_var_event ( void )`

Devuelve el evento de allegro.

Returns

ALLEGRO_EVENT

Definition at line 418 of file allegro_stuff.c.

4.16.4.13 allegro_get_var_font() `ALLEGRO_FONT * allegro_get_var_font ( void )`

Devuelve la fuente de allegro.

Returns

ALLEGRO_FONT

Definition at line 443 of file allegro_stuff.c.

4.16.4.14 allegro_get_var_font_h() `int allegro_get_var_font_h ( void )`

Devuelve el alto de un caracter de la funete usada.

Returns

int alto

Definition at line 448 of file allegro_stuff.c.

4.16.4.15 allegro_get_var_font_w() `int allegro_get_var_font_w ( void )`

Devuelve ancho de un caracter de la fuente usada.

Returns

int ancho

Definition at line 453 of file allegro_stuff.c.

4.16.4.16 allegro_get_var_redraw() `bool allegro_get_var_redraw ( void )`

Devuelve flag de renderización.

Returns

true Renderiza

false No renderiza

Definition at line 428 of file allegro_stuff.c.

4.16.4.17 allegro_inits() `void allegro_inits ( void )`

Inicializaciones de allegro. Si algo falla, lo notifica por consola y finaliza el programa.

Definition at line 285 of file allegro_stuff.c.

4.16.4.18 allegro_is_event_queueVacia() bool allegro_is_event_queueVacia (void)

Informa si al cola de eventos está vacía o no.

Returns

true Vacía

false No vacía

Definition at line 473 of file allegro_stuff.c.

4.16.4.19 allegro_reinit_display() void allegro_reinit_display (void)

Reinicializa el display de allegro.

Definition at line 349 of file allegro_stuff.c.

4.16.4.20 allegro_rick_draw() void allegro_rick_draw (void)

Definition at line 773 of file allegro_stuff.c.

4.16.4.21 allegro_rick_off() void allegro_rick_off (void)

Definition at line 766 of file allegro_stuff.c.

4.16.4.22 allegro_rick_on() void allegro_rick_on (void)

Definition at line 748 of file allegro_stuff.c.

4.16.4.23 allegro_set_last_key() void allegro_set_last_key (unsigned char *allegro_key_code*)

Setea una ultima tecla presionada.

Parameters

allegro_key_code	ALLEGRO_KEY_CODE

Definition at line 396 of file allegro_stuff.c.

4.16.4.24 allegro_set_rick_flag() `void allegro_set_rick_flag ( bool state )`

Parameters

state	

Definition at line 761 of file allegro_stuff.c.

4.16.4.25 allegro_set_var_done() `void allegro_set_var_done ( bool state )`

Setea flag de finalización del programa.

Parameters

state	true or false

Definition at line 433 of file allegro_stuff.c.

4.16.4.26 allegro_set_var_event() `void allegro_set_var_event ( ALLEGRO_EVENT event )`

Carga un evento de allegro.

Parameters

event	

Definition at line 483 of file allegro_stuff.c.

4.16.4.27 allegro_set_var_redraw() `void allegro_set_var_redraw ( bool state )`

Setea flag de renderizacion.

Parameters

state	true or false

Definition at line 438 of file allegro_stuff.c.

4.16.4.28 allegro_sound_pause_stream() `void allegro_sound_pause_stream ( void )`

Pausa la musica actual, si hay alguna seleccionada.

Definition at line 631 of file allegro_stuff.c.

4.16.4.29 allegro_sound_play_effect_bonus() `void allegro_sound_play_effect_bonus ( void )`

Reproduce efecto de bonus.

Definition at line 669 of file allegro_stuff.c.

4.16.4.30 allegro_sound_play_effect_click() `void allegro_sound_play_effect_click ( void )`

Reproduce efecto de click (seleccion de menu//aceptar//etc.)

Definition at line 674 of file allegro_stuff.c.

4.16.4.31 allegro_sound_play_effect_coin_drop() `void allegro_sound_play_effect_coin_drop ( void )`

Reproduce efecto `de moneda tirada`

Definition at line 729 of file allegro_stuff.c.

4.16.4.32 allegro_sound_play_effect_crash() `void allegro_sound_play_effect_crash ( void )`

Reproduce efecto de choque.

Definition at line 679 of file allegro_stuff.c.

4.16.4.33 allegro_sound_play_effect_drowned() void allegro_sound_play_effect_drowned (void)

Reproduce efecto de ahogado (toco agua)

Definition at line 684 of file allegro_stuff.c.

4.16.4.34 allegro_sound_play_effect_exiting() void allegro_sound_play_effect_exiting (void)

Reproduce efecto de 'saliendo'.

Definition at line 719 of file allegro_stuff.c.

4.16.4.35 allegro_sound_play_effect_goal() void allegro_sound_play_effect_goal (void)

Reproduce efecto de 'llego a la meta'.

Definition at line 689 of file allegro_stuff.c.

4.16.4.36 allegro_sound_play_effect_jump() void allegro_sound_play_effect_jump (void)

Reproduce efecto de salto.

Definition at line 694 of file allegro_stuff.c.

4.16.4.37 allegro_sound_play_effect_low_time() void allegro_sound_play_effect_low_time (void)

Reproduce efecto de 'queda poco tiempo'.

Definition at line 699 of file allegro_stuff.c.

4.16.4.38 allegro_sound_play_effect_menu_enter() void allegro_sound_play_effect_menu_enter (void)

Reproduce efecto de 'menu enter'.

Definition at line 709 of file allegro_stuff.c.

4.16.4.39 allegro_sound_play_effect_new_max_score() `void allegro_sound_play_effect_new_max_score (`

```
            void  )
```

Reproduce efecto de 'new_max_score'.

Definition at line 714 of file allegro_stuff.c.

4.16.4.40 allegro_sound_play_effect_no_time() `void allegro_sound_play_effect_no_time (`

```
            void  )
```

Reproduce efecto de 'sin tiempo'.

Definition at line 724 of file allegro_stuff.c.

4.16.4.41 allegro_sound_play_effect_run_completed() `void allegro_sound_play_effect_run_completed (`

```
            void  )
```

Reproduce efecto de 'run completada' (llego 5 veces a la meta)

Definition at line 704 of file allegro_stuff.c.

4.16.4.42 allegro_sound_play_stream() `void allegro_sound_play_stream (`

```
            void  )
```

Reproduce la musica actual, si hay alguna seleccionada.

Definition at line 622 of file allegro_stuff.c.

4.16.4.43 allegro_sound_restart_stream() `void allegro_sound_restart_stream (`

```
            void  )
```

Reinicia la musica actual, si hay alguna seleccionada.

Definition at line 640 of file allegro_stuff.c.

4.16.4.44 allegro_sound_set_stream_credits() `void allegro_sound_set_stream_credits (`

```
            void  )
```

Selecciona musica de creditos. Comienza pausada.

Definition at line 491 of file allegro_stuff.c.

4.16.4.45 allegro_sound_set_stream_gain_down() `void allegro_sound_set_stream_gain_down ( void )`

Reduce en 0.1 la ganancia de stream.

Definition at line 657 of file allegro_stuff.c.

4.16.4.46 allegro_sound_set_stream_gain_up() `void allegro_sound_set_stream_gain_up ( void )`

Aumenta en 0.1 la ganancia de stream.

Definition at line 648 of file allegro_stuff.c.

4.16.4.47 allegro_sound_set_stream_game_over() `void allegro_sound_set_stream_game_over ( void )`

Selecciona musica de game over. Comienza pausada.

Definition at line 587 of file allegro_stuff.c.

4.16.4.48 allegro_sound_set_stream_main_menu() `void allegro_sound_set_stream_main_menu ( void )`

Selecciona musica de menu. Comienza pausada.

Definition at line 507 of file allegro_stuff.c.

4.16.4.49 allegro_sound_set_stream_pause_menu() `void allegro_sound_set_stream_pause_menu ( void )`

Selecciona musica de pausa. Comienza pausada.

Definition at line 523 of file allegro_stuff.c.

4.16.4.50 allegro_sound_set_stream_playing() `void allegro_sound_set_stream_playing ( void )`

Selecciona musica de jugando. Comienza pausada.

Definition at line 555 of file allegro_stuff.c.

4.16.4.51 allegro_sound_set_stream_ranking() `void allegro_sound_set_stream_ranking (`

```
void )
```

Selecciona musica de ranking. Comienza pausada.

Definition at line 539 of file allegro_stuff.c.

4.16.4.52 allegro_sound_set_stream_rick() `void allegro_sound_set_stream_rick (`

```
void )
```

Definition at line 571 of file allegro_stuff.c.

4.16.4.53 allegro_sound_toggle_stream() `void allegro_sound_toggle_stream (`

```
void )
```

Cambia estado de reproduccion de la musica actual, si hay alguna seleccionada.

Definition at line 606 of file allegro_stuff.c.

4.16.4.54 allegro_wait_for_event() `ALLEGRO_EVENT_TYPE allegro_wait_for_event (`

```
void )
```

Espera a que ocurra un evento y lo devuelve.

Returns

ALLEGRO_EVENT_TYPE

Definition at line 401 of file allegro_stuff.c.

4.16.4.55 must_init() `void must_init (`

```
bool test,
const char * description )
```

Inicializa "cosas", y sale del programa si hay problemas, avisando dónde estuvo.

Parameters

test	Handler//booleano con status de la inicialización.
description	String con la descripción/nombre de la "cosa" a inicializar.

Definition at line 275 of file allegro_stuff.c.

4.16.5 Variable Documentation

4.16.5.1 sprites `sprites_t sprites`

Definition at line 151 of file allegro_stuff.c.

4.17 allegro_stuff.c

Go to the documentation of this file.

```
00001
00013 /*******************************************************************************
00014  * INCLUDE HEADER FILES
00015  ******************************************************************************/
00016
00017 #include "allegro_stuff.h"
00018 #include "geometry.h"
00019 #include <string.h>
00020
00021 #include "./algif5/algif.h"
00022
00023 /*******************************************************************************
00024  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00025  ******************************************************************************/
00026
00027 // Altura de la fuente
00028 #define FONT_HEIGHT 16
00029
00030 // Nombres de los stream files
00031 #define SOUND_STREAM_FILE_CREDITS "credits_theme"
00032 #define SOUND_STREAM_FILE_MAIN "main_menu_theme"
00033 #define SOUND_STREAM_FILE_PAUSE "pause_menu_theme"
00034 #define SOUND_STREAM_FILE_PLAYING "playing_theme"
00035 #define SOUND_STREAM_FILE_RANKING "ranking_theme"
00036 #define SOUND_STREAM_FILE_RICK "rick"
00037 #define SOUND_STREAM_FILE_GAME_OVER "game_over"
00038
00039 #define FONT_FILE_NAME "PublicPixel.ttf"
00040
00041 // Nombres de los sprites
00042 #define SPRITE_HEART "minecraft_heart"
00043 #define SPRITE_BACKGROUND "sprite_background"
00044 #define SPRITE_CAR "sprite_cars"
00045 #define SPRITE_FROG "sprite_frog"
00046 #define SPRITE_LOG "sprite_log"
00047 #define SPRITE_TURTLES "sprite_turtles"
00048 #define SPRITE_MENU_HOME_BACK "sprite_menu_home_background"
00049 #define SPRITE_MENU_HOME "sprite_menu_home"
00050 /*
00051 #define SPRITE_MENU_DIFF_BACK       "sprite_menu_home_background"
00052 #define SPRITE_MENU_DIFF            "sprite_menu_home"
00053 #define SPRITE_MENU_PAUSE_BACK      "sprite_menu_home_background"
00054 #define SPRITE_MENU_PAUSE           "sprite_menu_home"
00055 */
00056 #define SPRITE_MENU_DIFF_BACK "sprite_menu_diff_background"
00057 #define SPRITE_MENU_DIFF "sprite_menu_diff"
00058 #define SPRITE_MENU_PAUSE_BACK "sprite_menu_pause_background"
00059 #define SPRITE_MENU_PAUSE "sprite_menu_pause"
00060 #define SPRITE_MENU_GAME_OVER_BACK "sprite_menu_gameover_background"
00061 #define SPRITE_MENU_GAME_OVER "sprite_menu_gameover"
00062 #define SPRITE_CREDITS "sprite_credits"
00063 #define SPRITE_NAME "sprite_name"
00064 #define SPRITE_ICON "icon"
00065 #define SPRITE_DEAD "sprite_dead"
00066 #define SPRITE_BORDER "sprite_border"
00067 #define SPRITE_SPLASH "sprite_splash"
00068 #define SPRITE_COIN "sprite_coin"
00069
00070 // Extensiones
00071 #define EXTENSION_SOUND_SAMPLE ".wav"
00072 #define EXTENSION_SOUND_STREAM ".opus"
00073 #define EXTENSION_SPRITES ".png"
00074
00075 // Local paths
00076 #define PATH_SOUND_STREAMS "../res/sounds/streams/"
00077 #define PATH_SOUND_SAMPLES "../res/sounds/samples/"
```

```
00078 #define PATH_FONTS "../res/fonts/"
00079 #define PATH_SPRITES "../res/sprites/"
00080 #define PATH_GIFS "../res/gifs/"
00081
00082 #define GLOBAL_STREAM_VOLUME (double)0.5
00083
00084 /*****************************************************************************
00085  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00086  *****************************************************************************/
00087
00088 typedef struct
00089 {
00091     ALLEGRO_TIMER *timer;
00092
00093     // cola de eventos
00094     ALLEGRO_EVENT_QUEUE *queue;
00095
00096     // display
00097     ALLEGRO_DISPLAY *disp;
00098
00099     // fuente builtin
00100     ALLEGRO_FONT *font;
00101     int font_h; // altura de un caracter
00102     int font_w; // ancho de un caracter
00103
00104     // variable evento
00105     ALLEGRO_EVENT event;
00106
00107     // flag para salir el programa
00108     bool done;
00109     // flag para renderizar
00110     bool redraw;
00111
00112 } allegro_t;
00113
00114 typedef struct
00115 {
00116     ALLEGRO_AUDIO_STREAM *stream;
00117     unsigned char stream_state;
00118
00119     struct
00120     {
00121         ALLEGRO_SAMPLE *jump;
00122         ALLEGRO_SAMPLE *crash;
00123         ALLEGRO_SAMPLE *goal;
00124         ALLEGRO_SAMPLE *low_time;
00125         ALLEGRO_SAMPLE *click;
00126         ALLEGRO_SAMPLE *bonus;
00127         ALLEGRO_SAMPLE *run_completed;
00128         ALLEGRO_SAMPLE *drowned;
00129         ALLEGRO_SAMPLE *menu_enter;
00130         ALLEGRO_SAMPLE *new_max_score;
00131         ALLEGRO_SAMPLE *exiting;
00132         ALLEGRO_SAMPLE *no_time;
00133         ALLEGRO_SAMPLE *coin_drop;
00134     } samples;
00135
00136 } sounds_t;
00137
00138 enum SOUND_STREAM_STATES
00139 {
00140     SOUND_STREAM_STATE_NO_INIT,
00141     SOUND_STREAM_STATE_INIT,
00142     SOUND_STREAM_STATE_PAUSE,
00143     SOUND_STREAM_STATE_PLAY
00144 };
00145
00146 /*****************************************************************************
00147  * VARIABLES WITH GLOBAL SCOPE
00148  *****************************************************************************/
00149
00150 // estructura con punteros a sprites
00151 sprites_t sprites;
00152
00153 /*****************************************************************************
00154  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00155  *****************************************************************************/
00156
00168 static ALLEGRO_BITMAP *sprite_cut(ALLEGRO_BITMAP *source_bmp, int x, int y, int w, int h);
00169
00174 static void sprites_init(void);
00175
00191 static char *make_sprite_path(char *file_name, char *prev_str);
00192
00197 static void sprites_deinit(void);
00198
00203 static void audio_init(void);
```

```
00204
00209 static void audio_deinit(void);
00210
00219 static bool init_audio_stream(const char *file, float gain);
00220
00229 static bool init_sample(ALLEGRO_SAMPLE **sample, const char *file);
00230
00235 static void rick_init(void);
00236
00241 static void rick_deinit(void);
00242
00243 /*****************************************************************************
00244  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00245  *****************************************************************************/
00246
00247 // variables principales de allegro
00248 static allegro_t allegro_vars;
00249
00250 // Ultima tecla presionada
00251 static unsigned char last_key;
00252
00253 // variable con los sonidos/musicas del juego
00254 static sounds_t sounds;
00255
00256 // nombre del ultimo stream inicializado
00257 static char last_init_stream[30];
00258
00259 static ALGIF_ANIMATION *rick;
00260
00261 static char rick_prev_stream[30];
00262
00263 static bool rick_flag;
00264
00265 static ALLEGRO_MONITOR_INFO monitor_info;
00266
00267 static double stream_gain = GLOBAL_STREAM_VOLUME;
00268
00269 /*****************************************************************************
00270  *****************************************************************************
00271             GLOBAL FUNCTION DEFINITIONS
00272  *****************************************************************************
00273  *****************************************************************************/
00274
00275 void must_init(bool test, const char *description)
00276 {
00277     if (!test)
00278     {
00279         printf("~no se pudo inicializar %s~\n", description);
00280         while (true)
00281             ;
00282     }
00283 }
00284
00285 void allegro_inits(void)
00286 {
00287     must_init(al_init(), "allegro");
00288     must_init(al_install_keyboard(), "keyboard");
00289     must_init(al_install_mouse(), "mouse");
00290     must_init(al_init_image_addon(), "image");
00291     al_init_font_addon();
00292     must_init(al_init_ttf_addon(), "ttf addon");
00293
00294     // timer que actualiza cada 1/60 segundos (60fps)
00295     allegro_vars.timer = al_create_timer(1.0 / FPS);
00296     must_init(allegro_vars.timer, "timer");
00297
00298     // cola de eventos
00299     allegro_vars.queue = al_create_event_queue();
00300     must_init(allegro_vars.queue, "queue");
00301
00302     // Inicializa los spritesheets.
00303     sprites_init();
00304
00305     // para dibujar figuras primitivas (círculos, rectángulos, líneas, rellenos o no, etc.)
00306     must_init(al_init_primitives_addon(), "primitives");
00307
00308     // registra eventos posibles
00309     al_register_event_source(allegro_vars.queue, al_get_keyboard_event_source());
00310     al_register_event_source(allegro_vars.queue, al_get_timer_event_source(allegro_vars.timer));
00311     al_register_event_source(allegro_vars.queue, al_get_mouse_event_source());
00312
00313     // flag para salir el programa
00314     allegro_vars.done = false;
00315     // flag para renderizar
00316     allegro_vars.redraw = false;
00317
00318     // audio
```

```
00319     must_init(al_install_audio(), "audio");
00320     must_init(al_init_acodec_addon(), "audio codecs");
00321     must_init(al_reserve_samples(16), "reserve samples");
00322 
00323     audio_init();
00324 
00325     rick_init();
00326 
00327     must_init(al_get_monitor_info(0, &monitor_info), "getting monitor info");
00328 
00329     // creacion del display
00330     allegro_reinit_display();
00331 
00332     // inicializa timer
00333     al_start_timer(allegro_vars.timer);
00334 }
00335 
00336 void allegro_deinits(void)
00337 {
00338     rick_deinit();
00339     sprites_deinit();
00340     audio_deinit();
00341     al_destroy_font(allegro_vars.font);
00342 
00343     // al_destroy_display(allegro_vars.disp);
00344 
00345     al_destroy_timer(allegro_vars.timer);
00346     al_destroy_event_queue(allegro_vars.queue);
00347 }
00348 
00349 void allegro_reinit_display(void)
00350 {
00351 
00352     al_set_new_bitmap_flags(ALLEGRO_MIN_LINEAR | ALLEGRO_MAG_LINEAR);
00353     // Para tener aceleracion por HW desde la GPU (hace que no explote con los draw_text)
00354     al_set_new_bitmap_flags(ALLEGRO_VIDEO_BITMAP);
00355     // al_set_new_display_flags(ALLEGRO_RESIZABLE);
00356 
00357     // Titulo de la ventana
00358     al_set_new_window_title("~ Programación I ~ TP Final ~ Frogger ~");
00359     // Centrado en pantalla, según el monitor
00360     al_set_new_window_position(monitor_info.x2 / 2 - DISPLAY_W / 2, monitor_info.y2 / 2 - DISPLAY_H /
      2 - 50);
00361 
00362     // opciones para el display (antialiasing)
00363     al_set_new_display_option(ALLEGRO_SAMPLE_BUFFERS, 1, ALLEGRO_SUGGEST);
00364     al_set_new_display_option(ALLEGRO_SAMPLES, 8, ALLEGRO_SUGGEST);
00365 
00366     // creación del display
00367     allegro_vars.disp = al_create_display(DISPLAY_W, DISPLAY_H);
00368     must_init(allegro_vars.disp, "display");
00369     al_register_event_source(allegro_vars.queue, al_get_display_event_source(allegro_vars.disp));
00370 
00371     al_set_display_icon(allegro_vars.disp, sprites.icon);
00372 
00373     // Reload de la fuente
00374     char string[60] = PATH_FONTS;
00375     strcat(string, FONT_FILE_NAME);
00376     allegro_vars.font = al_load_font(string, FONT_HEIGHT, 0);
00377     must_init(allegro_vars.font, "font");
00378     allegro_vars.font_h = al_get_font_line_height(allegro_vars.font);
00379     allegro_vars.font_w = al_get_text_width(allegro_vars.font, "a");
00380 }
00381 
00382 void allegro_deinit_display(void)
00383 {
00384     if (allegro_vars.disp != NULL)
00385     {
00386         al_unregister_event_source(allegro_vars.queue,
      al_get_display_event_source(allegro_vars.disp));
00387         al_destroy_display(allegro_vars.disp);
00388     }
00389 }
00390 
00391 unsigned char allegro_get_last_key(void)
00392 {
00393     return (last_key);
00394 }
00395 
00396 void allegro_set_last_key(unsigned char allegro_key_code)
00397 {
00398     last_key = allegro_key_code;
00399 }
00400 
00401 ALLEGRO_EVENT_TYPE allegro_wait_for_event(void)
00402 {
00403     al_wait_for_event(allegro_vars.queue, &allegro_vars.event);
```

```
00404
00405     return (allegro_vars.event.type);
00406 }
00407
00408 ALLEGRO_EVENT *allegro_get_next_event(void)
00409 {
00410     bool flag = al_get_next_event(allegro_vars.queue, &allegro_vars.event);
00411
00412     if (flag)
00413         return (&allegro_vars.event);
00414     else
00415         return NULL;
00416 }
00417
00418 ALLEGRO_EVENT allegro_get_var_event(void)
00419 {
00420     return (allegro_vars.event);
00421 }
00422
00423 bool allegro_get_var_done(void)
00424 {
00425     return (allegro_vars.done);
00426 }
00427
00428 bool allegro_get_var_redraw(void)
00429 {
00430     return (allegro_vars.redraw);
00431 }
00432
00433 void allegro_set_var_done(bool state)
00434 {
00435     allegro_vars.done = state;
00436 }
00437
00438 void allegro_set_var_redraw(bool state)
00439 {
00440     allegro_vars.redraw = state;
00441 }
00442
00443 ALLEGRO_FONT *allegro_get_var_font(void)
00444 {
00445     return (allegro_vars.font);
00446 }
00447
00448 int allegro_get_var_font_h(void)
00449 {
00450     return (allegro_vars.font_h);
00451 }
00452
00453 int allegro_get_var_font_w(void)
00454 {
00455     return (allegro_vars.font_w);
00456 }
00457
00458 void allegro_clear_display(void)
00459 {
00460     al_clear_to_color(al_map_rgb(0, 0, 0));
00461 }
00462
00463 void allegro_draw_background(void)
00464 {
00465     al_draw_bitmap(sprites.background, 0, 0, 0);
00466 }
00467
00468 void allegro_draw_menu_background(int window)
00469 {
00470     al_draw_bitmap(sprites.menu[window].background, 0, 0, 0);
00471 }
00472
00473 bool allegro_is_event_queueVacia(void)
00474 {
00475     return (al_is_event_queue_empty(allegro_vars.queue));
00476 }
00477
00478 ALLEGRO_EVENT_QUEUE *allegro_get_event_queue(void)
00479 {
00480     return (allegro_vars.queue);
00481 }
00482
00483 void allegro_set_var_event(ALLEGRO_EVENT event)
00484 {
00485     allegro_vars.event = event;
00486 }
00487
00488 #pragma region allegro_sound
00489
00490 #pragma region allegro_sound_set_stream
```

```
00491 void allegro_sound_set_stream_credits(void)
00492 {
00493     char file[] = SOUND_STREAM_FILE_CREDITS;
00494 
00495     // si ya estaba inicializado...
00496     if (strcmp(file, last_init_stream) == 0)
00497     {
00498         allegro_sound_pause_stream();
00499     }
00500     else
00501     {
00502         must_init(init_audio_stream(file, stream_gain),
00503                   "credits stream");
00504     }
00505 }
00506 
00507 void allegro_sound_set_stream_main_menu(void)
00508 {
00509     char file[] = SOUND_STREAM_FILE_MAIN;
00510 
00511     // si ya estaba inicializado...
00512     if (strcmp(file, last_init_stream) == 0)
00513     {
00514         allegro_sound_pause_stream();
00515     }
00516     else
00517     {
00518         must_init(init_audio_stream(file, stream_gain),
00519                   "main menu stream");
00520     }
00521 }
00522 
00523 void allegro_sound_set_stream_pause_menu(void)
00524 {
00525     char file[] = SOUND_STREAM_FILE_PAUSE;
00526 
00527     // si ya estaba inicializado...
00528     if (strcmp(file, last_init_stream) == 0)
00529     {
00530         allegro_sound_pause_stream();
00531     }
00532     else
00533     {
00534         must_init(init_audio_stream(file, stream_gain),
00535                   "pause menu stream");
00536     }
00537 }
00538 
00539 void allegro_sound_set_stream_ranking(void)
00540 {
00541     char file[] = SOUND_STREAM_FILE_RANKING;
00542 
00543     // si ya estaba inicializado...
00544     if (strcmp(file, last_init_stream) == 0)
00545     {
00546         allegro_sound_pause_stream();
00547     }
00548     else
00549     {
00550         must_init(init_audio_stream(file, stream_gain),
00551                   "ranking stream");
00552     }
00553 }
00554 
00555 void allegro_sound_set_stream_playing(void)
00556 {
00557     char file[] = SOUND_STREAM_FILE_PLAYING;
00558 
00559     // si ya estaba inicializado...
00560     if (strcmp(file, last_init_stream) == 0)
00561     {
00562         allegro_sound_pause_stream();
00563     }
00564     else
00565     {
00566         must_init(init_audio_stream(file, stream_gain),
00567                   "playing stream");
00568     }
00569 }
00570 
00571 void allegro_sound_set_stream_rick(void)
00572 {
00573     char file[] = SOUND_STREAM_FILE_RICK;
00574 
00575     // si ya estaba inicializado...
00576     if (strcmp(file, last_init_stream) == 0)
00577     {
```

```
00578         allegro_sound_pause_stream();
00579     }
00580     else
00581     {
00582         must_init(init_audio_stream(file, stream_gain),
00583                   "credtis stream");
00584     }
00585 }
00586
00587 void allegro_sound_set_stream_game_over(void)
00588 {
00589     char file[] = SOUND_STREAM_FILE_GAME_OVER;
00590
00591     // si ya estaba inicializado...
00592     if (strcmp(file, last_init_stream) == 0)
00593     {
00594         allegro_sound_pause_stream();
00595     }
00596     else
00597     {
00598         must_init(init_audio_stream(file, stream_gain),
00599                   "game over stream");
00600     }
00601 }
00602
00603 #pragma endregion allegro_sound_set_stream
00604
00605 #pragma region allegro_sound_control
00606 void allegro_sound_toggle_stream(void)
00607 {
00608     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00609     {
00610         bool state = al_get_audio_stream_playing(sounds.stream);
00611
00612         must_init(al_set_audio_stream_playing(sounds.stream, !state),
00613                   "set to toggle stream");
00614
00615         if (!state)
00616             sounds.stream_state = SOUND_STREAM_STATE_PAUSE;
00617         else
00618             sounds.stream_state = SOUND_STREAM_STATE_PLAY;
00619     }
00620 }
00621
00622 void allegro_sound_play_stream(void)
00623 {
00624     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00625     {
00626         al_set_audio_stream_playing(sounds.stream, true);
00627         sounds.stream_state = SOUND_STREAM_STATE_PLAY;
00628     }
00629 }
00630
00631 void allegro_sound_pause_stream(void)
00632 {
00633     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00634     {
00635         al_set_audio_stream_playing(sounds.stream, false);
00636         sounds.stream_state = SOUND_STREAM_STATE_PAUSE;
00637     }
00638 }
00639
00640 void allegro_sound_restart_stream(void)
00641 {
00642     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00643     {
00644         init_audio_stream(last_init_stream, stream_gain);
00645     }
00646 }
00647
00648 void allegro_sound_set_stream_gain_up(void)
00649 {
00650     if (stream_gain <= 0.9)
00651     {
00652         stream_gain += 0.1;
00653         al_set_audio_stream_gain(sounds.stream, stream_gain);
00654     }
00655 }
00656
00657 void allegro_sound_set_stream_gain_down(void)
00658 {
00659     if (stream_gain >= 0.1)
00660     {
00661         stream_gain -= 0.1;
00662         al_set_audio_stream_gain(sounds.stream, stream_gain);
00663     }
00664 }
```

```
00665
00666 #pragma endregion allegro_sound_control
00667
00668 #pragma region allegro_sound_play_sample
00669 void allegro_sound_play_effect_bonus(void)
00670 {
00671     al_play_sample(sounds.samples.bonus, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00672 }
00673
00674 void allegro_sound_play_effect_click(void)
00675 {
00676     al_play_sample(sounds.samples.click, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00677 }
00678
00679 void allegro_sound_play_effect_crash(void)
00680 {
00681     al_play_sample(sounds.samples.crash, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00682 }
00683
00684 void allegro_sound_play_effect_drowned(void)
00685 {
00686     al_play_sample(sounds.samples.drowned, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00687 }
00688
00689 void allegro_sound_play_effect_goal(void)
00690 {
00691     al_play_sample(sounds.samples.goal, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00692 }
00693
00694 void allegro_sound_play_effect_jump(void)
00695 {
00696     al_play_sample(sounds.samples.jump, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00697 }
00698
00699 void allegro_sound_play_effect_low_time(void)
00700 {
00701     al_play_sample(sounds.samples.low_time, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00702 }
00703
00704 void allegro_sound_play_effect_run_completed(void)
00705 {
00706     al_play_sample(sounds.samples.run_completed, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00707 }
00708
00709 void allegro_sound_play_effect_menu_enter(void)
00710 {
00711     al_play_sample(sounds.samples.menu_enter, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00712 }
00713
00714 void allegro_sound_play_effect_new_max_score(void)
00715 {
00716     al_play_sample(sounds.samples.new_max_score, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00717 }
00718
00719 void allegro_sound_play_effect_exiting(void)
00720 {
00721     al_play_sample(sounds.samples.exiting, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00722 }
00723
00724 void allegro_sound_play_effect_no_time(void)
00725 {
00726     al_play_sample(sounds.samples.no_time, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00727 }
00728
00729 void allegro_sound_play_effect_coin_drop(void)
00730 {
00731     al_play_sample(sounds.samples.coin_drop, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00732 }
00733
00734 #pragma endregion allegro_sound_play_sample
00735
00736 #pragma endregion allegro_sound
00737
00738 void allegro_draw_hitbox(int x, int y, int w, int h)
00739 {
00740     al_draw_rectangle(x,
00741                       y,
00742                       x + w,
00743                       y + h,
00744                       al_map_rgb(100, 100, 100),
00745                       1); // grosor
00746 }
00747
00748 void allegro_rick_on(void)
00749 {
00750     strcpy(rick_prev_stream, last_init_stream);
00751
```

```
00752     allegro_sound_set_stream_rick();
00753     allegro_sound_play_stream();
00754 }
00755
00756 bool allegro_get_rick_flag(void)
00757 {
00758     return rick_flag;
00759 }
00760
00761 void allegro_set_rick_flag(bool state)
00762 {
00763     rick_flag = state;
00764 }
00765
00766 void allegro_rick_off(void)
00767 {
00768     must_init(init_audio_stream(rick_prev_stream, stream_gain),
00769               "retornando stream ~~ sacando rick");
00770     allegro_sound_play_stream();
00771 }
00772
00773 void allegro_rick_draw(void)
00774 {
00775     al_draw_bitmap(algif_get_bitmap(rick, al_get_time()), 100, DISPLAY_H / 2, 0);
00776 }
00777
00778 /*****************************************************************************
00779  *****************************************************************************
00780             LOCAL FUNCTION DEFINITIONS
00781  *****************************************************************************
00782  *****************************************************************************/
00783
00784 static ALLEGRO_BITMAP *sprite_cut(ALLEGRO_BITMAP *source_bmp, int x, int y, int w, int h)
00785 {
00786     ALLEGRO_BITMAP *sprite = al_create_sub_bitmap(source_bmp, x, y, w, h);
00787     must_init(sprite, "sprite cut");
00788     return sprite;
00789 }
00790
00791 static void sprites_init(void)
00792 {
00793     int i, offset;
00794     pair_xy_t temp_xy;
00795     int temp_w, temp_h;
00796
00797     char *path = NULL;
00798
00799     // de la rana completo
00800     path = make_sprite_path(SPRITE_FROG, NULL);
00801     sprites.frog_uncut = al_load_bitmap(path);
00802
00803     // se particiona el de la rana en sus 8 partes
00804     for (i = 0; i < FROG_FRAMES; i++)
00805     {
00806         temp_xy = getXYFromFrogFrame(i);
00807
00808         if (!(i % 2)) // los sprites pares
00809         {
00810             temp_w = SPRITE_SIZE_FROG_STATIC_W;
00811             temp_h = SPRITE_SIZE_FROG_STATIC_H;
00812         }
00813         else if (i == 1 || i == 7)
00814         {
00815             temp_w = SPRITE_SIZE_FROG_DYNAMIC_SHORT;
00816             temp_h = SPRITE_SIZE_FROG_DYNAMIC_LONG;
00817         }
00818         else
00819         {
00820             temp_w = SPRITE_SIZE_FROG_DYNAMIC_LONG;
00821             temp_h = SPRITE_SIZE_FROG_DYNAMIC_SHORT;
00822         }
00823
00824         sprites.frog[i] = sprite_cut(sprites.frog_uncut, temp_xy.x, temp_xy.y, temp_w, temp_h);
00825     }
00826
00827     // el del fondo
00828     path = make_sprite_path(SPRITE_BACKGROUND, path);
00829     sprites.background = al_load_bitmap(path);
00830
00831     // el de los troncos
00832     path = make_sprite_path(SPRITE_LOG, path);
00833     sprites.log = al_load_bitmap(path);
00834
00835     // recorte de los autos.
00836
00837     path = make_sprite_path(SPRITE_CAR, path);
00838     sprites.cars_uncut = al_load_bitmap(path);
```

```
00839
00840     sprites.car[0] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(0).x, getXYFromCarFrame(0).y,
      CAR_W, CAR_H);
00841     sprites.car[1] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(1).x, getXYFromCarFrame(1).y,
      CAR_W, CAR_H);
00842     sprites.car[2] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(2).x, getXYFromCarFrame(2).y,
      CAR_W, CAR_H);
00843     sprites.car[3] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(3).x, getXYFromCarFrame(3).y,
      CAR_TRUCK_FIRE_W, CAR_H);
00844     sprites.car[4] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(4).x, getXYFromCarFrame(4).y,
      CAR_TRUCK_W, CAR_H);
00845
00846     // el de las tortugas sin recortar
00847     path = make_sprite_path(SPRITE_TURTLES, path);
00848     sprites.turtle_uncut = al_load_bitmap(path);
00849
00850     // se recortan los de la tortuga en sus 11 partes
00851     for (i = 0; i < TURTLE_FRAMES; i++)
00852     {
00853         temp_xy = getXYFromTurtleFrame(i);
00854
00855         sprites.turtle[i] = sprite_cut(sprites.turtle_uncut, temp_xy.x, temp_xy.y, TURTLE_SIDE,
      TURTLE_SIDE);
00856     }
00857
00858     // corazon
00859     path = make_sprite_path(SPRITE_HEART, path);
00860     sprites.heart = al_load_bitmap(path);
00861
00862     // fondo de menu
00863     path = make_sprite_path(SPRITE_MENU_HOME_BACK, path);
00864     sprites.menu[MENU_WINDOW_HOME].background = al_load_bitmap(path);
00865
00866     // botones con highlight, sin recortar
00867     path = make_sprite_path(SPRITE_MENU_HOME, path);
00868     sprites.menu[MENU_WINDOW_HOME].uncut = al_load_bitmap(path);
00869
00870     // se recortan los highlight
00871     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00872     {
00873         sprites.menu[MENU_WINDOW_HOME].option[i] = sprite_cut(sprites.menu[MENU_WINDOW_HOME].uncut,
00874                                                               MENU_OPTION_TOPLEFT_X,
00875                                                               MENU_OPTION_TOPLEFT_Y + offset,
00876                                                               MENU_OPTION_W,
00877                                                               MENU_OPTION_H);
00878     }
00879
00880     path = make_sprite_path(SPRITE_MENU_DIFF_BACK, path);
00881     sprites.menu[MENU_WINDOW_DIFFICULTY].background = al_load_bitmap(path);
00882
00883     path = make_sprite_path(SPRITE_MENU_DIFF, path);
00884     sprites.menu[MENU_WINDOW_DIFFICULTY].uncut = al_load_bitmap(path);
00885
00886     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00887     {
00888         sprites.menu[MENU_WINDOW_DIFFICULTY].option[i] =
      sprite_cut(sprites.menu[MENU_WINDOW_DIFFICULTY].uncut,
00889                                                                 MENU_OPTION_TOPLEFT_X,
00890                                                                 MENU_OPTION_TOPLEFT_Y + offset,
00891                                                                 MENU_OPTION_W,
00892                                                                 MENU_OPTION_H);
00893     }
00894
00895     path = make_sprite_path(SPRITE_MENU_PAUSE_BACK, path);
00896     sprites.menu[MENU_WINDOW_PAUSE].background = al_load_bitmap(path);
00897
00898     path = make_sprite_path(SPRITE_MENU_PAUSE, path);
00899     sprites.menu[MENU_WINDOW_PAUSE].uncut = al_load_bitmap(path);
00900
00901     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00902     {
00903         sprites.menu[MENU_WINDOW_PAUSE].option[i] = sprite_cut(sprites.menu[MENU_WINDOW_PAUSE].uncut,
00904                                                               MENU_OPTION_TOPLEFT_X,
00905                                                               MENU_OPTION_TOPLEFT_Y + offset,
00906                                                               MENU_OPTION_W,
00907                                                               MENU_OPTION_H);
00908     }
00909
00910     path = make_sprite_path(SPRITE_MENU_GAME_OVER_BACK, path);
00911     sprites.menu[MENU_WINDOW_GAME_OVER].background = al_load_bitmap(path);
00912
00913     path = make_sprite_path(SPRITE_MENU_GAME_OVER, path);
00914     sprites.menu[MENU_WINDOW_GAME_OVER].uncut = al_load_bitmap(path);
00915
00916     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00917     {
00918         sprites.menu[MENU_WINDOW_GAME_OVER].option[i] =
```

```
    sprite_cut(sprites.menu[MENU_WINDOW_GAME_OVER].uncut,
00919                                                                   MENU_OPTION_TOPLEFT_X,
00920                                                                   MENU_OPTION_TOPLEFT_Y + offset,
00921                                                                   MENU_OPTION_W,
00922                                                                   MENU_OPTION_H);
00923     }
00924
00925     path = make_sprite_path(SPRITE_CREDITS, path);
00926     sprites.credits = al_load_bitmap(path);
00927
00928     path = make_sprite_path(SPRITE_NAME, path);
00929     sprites.name = al_load_bitmap(path);
00930
00931     path = make_sprite_path(SPRITE_ICON, path);
00932     sprites.icon = al_load_bitmap(path);
00933
00934     path = make_sprite_path(SPRITE_DEAD, path);
00935     sprites.dead = al_load_bitmap(path);
00936
00937     path = make_sprite_path(SPRITE_COIN, path);
00938     sprites.coin.uncut = al_load_bitmap(path);
00939     for (i = 0; i < SPRITE_COIN_FRAMES; i++)
00940     {
00941         pair_xy_t coord = getXYFromCoinFrame(i);
00942         sprites.coin.frame[i] = sprite_cut(sprites.coin.uncut, coord.x, coord.y, SPRITE_COIN_SIDE,
    SPRITE_COIN_SIDE);
00943     }
00944
00945     path = make_sprite_path(SPRITE_SPLASH, path);
00946     sprites.splash.uncut = al_load_bitmap(path);
00947     for (i = 0; i < SPRITE_SPLASH_FRAMES; i++)
00948     {
00949         pair_xy_t coord = getXYFromSplashFrame(i);
00950         sprites.splash.frame[i] = sprite_cut(sprites.splash.uncut, coord.x, coord.y, SPRITE_SPLASH_W,
    SPRITE_SPLASH_H);
00951     }
00952
00953     path = make_sprite_path(SPRITE_BORDER, path);
00954     sprites.border = al_load_bitmap(path);
00955
00956     free(path);
00957 }
00958
00959 static char *make_sprite_path(char *file_name, char *prev_str)
00960 {
00961     if (prev_str != NULL)
00962         free(prev_str);
00963
00964     char *path = NULL;
00965
00966     int str_size = sizeof(PATH_SPRITES) + strlen(file_name) + sizeof(EXTENSION_SPRITES) + 1;
00967
00968     path = malloc(str_size);
00969     must_init(path, file_name);
00970
00971     memset(path, 0, str_size);
00972
00973     strcat(path, PATH_SPRITES);
00974     strcat(path, file_name);
00975     strcat(path, EXTENSION_SPRITES);
00976
00977     return path;
00978 }
00979
00980 static void sprites_deinit(void)
00981 {
00982     int i, j;
00983
00984     al_destroy_bitmap(sprites.frog_uncut);
00985
00986     for (i = 0; i < FROG_FRAMES; i++)
00987         al_destroy_bitmap(sprites.frog[i]);
00988
00989     al_destroy_bitmap(sprites.background);
00990
00991     al_destroy_bitmap(sprites.log);
00992
00993     for (i = 0; i < CAR_TYPE_N; i++)
00994         al_destroy_bitmap(sprites.car[i]);
00995
00996     al_destroy_bitmap(sprites.turtle_uncut);
00997
00998     for (i = 0; i < TURTLE_FRAMES; i++)
00999         al_destroy_bitmap(sprites.turtle[i]);
01000
01001     al_destroy_bitmap(sprites.heart);
01002
```

```
01003     for (i = 0; i < MENU_WINDOW_MAX; i++)
01004     {
01005         al_destroy_bitmap(sprites.menu[i].background);
01006         al_destroy_bitmap(sprites.menu[i].uncut);
01007 
01008         for (j = 0; j < MENU_STATE_MAX; j++)
01009         {
01010             al_destroy_bitmap(sprites.menu[i].option[j]);
01011         }
01012     }
01013 
01014     al_destroy_bitmap(sprites.credits);
01015 
01016     al_destroy_bitmap(sprites.name);
01017 
01018     al_destroy_bitmap(sprites.icon);
01019 
01020     al_destroy_bitmap(sprites.dead);
01021 
01022     al_destroy_bitmap(sprites.coin.uncut);
01023     for (i = 0; i < SPRITE_COIN_FRAMES; i++)
01024         al_destroy_bitmap(sprites.coin.frame[i]);
01025 
01026     al_destroy_bitmap(sprites.splash.uncut);
01027     for (i = 0; i < SPRITE_SPLASH_FRAMES; i++)
01028         al_destroy_bitmap(sprites.splash.frame[i]);
01029 
01030     al_destroy_bitmap(sprites.border);
01031 }
01032 
01033 static void audio_init(void)
01034 {
01035     // streams
01036     sounds.stream_state = SOUND_STREAM_STATE_NO_INIT;
01037 
01038     // efectos de sonido
01039     must_init(init_sample(&sounds.samples.bonus, "bonus_alert"),
01040               "effect_bonus sample");
01041 
01042     must_init(init_sample(&sounds.samples.click, "click"),
01043               "effect_click sample");
01044 
01045     must_init(init_sample(&sounds.samples.crash, "crash"),
01046               "effect_crash sample");
01047 
01048     must_init(init_sample(&sounds.samples.drowned, "fall_in_water"),
01049               "effect_drowned sample");
01050 
01051     must_init(init_sample(&sounds.samples.goal, "goal_reached"),
01052               "effect_goal sample");
01053 
01054     must_init(init_sample(&sounds.samples.jump, "jump_original"),
01055               "effect_jump sample");
01056 
01057     must_init(init_sample(&sounds.samples.low_time, "low_time_PC"),
01058               "effect_low_time sample");
01059 
01060     must_init(init_sample(&sounds.samples.run_completed, "run_completed"),
01061               "effect_run_completed sample");
01062 
01063     must_init(init_sample(&sounds.samples.menu_enter, "menu_enter"),
01064               "effect_menu_enter sample");
01065 
01066     must_init(init_sample(&sounds.samples.new_max_score, "new_max_score"),
01067               "effect_new_max_score sample");
01068 
01069     must_init(init_sample(&sounds.samples.exiting, "saliendo"),
01070               "effect_saliendo sample");
01071 
01072     must_init(init_sample(&sounds.samples.no_time, "no_time"),
01073               "effect_no_time sample");
01074 
01075     must_init(init_sample(&sounds.samples.coin_drop, "coin_drop"),
01076               "effect_coin_drop sample");
01077 }
01078 
01079 static void audio_deinit(void)
01080 {
01081     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
01082         al_destroy_audio_stream(sounds.stream);
01083 
01084     al_destroy_sample(sounds.samples.bonus);
01085     al_destroy_sample(sounds.samples.click);
01086     al_destroy_sample(sounds.samples.crash);
01087     al_destroy_sample(sounds.samples.drowned);
01088     al_destroy_sample(sounds.samples.goal);
01089     al_destroy_sample(sounds.samples.jump);
```

```
01090     al_destroy_sample(sounds.samples.low_time);
01091     al_destroy_sample(sounds.samples.run_completed);
01092     al_destroy_sample(sounds.samples.menu_enter);
01093     al_destroy_sample(sounds.samples.no_time);
01094     al_destroy_sample(sounds.samples.coin_drop);
01095 }
01096
01097 static bool init_audio_stream(const char *file, float gain)
01098 {
01099     if (file == NULL)
01100         return false;
01101
01102     ALLEGRO_AUDIO_STREAM **pt = &sounds.stream;
01103     unsigned char *state = &sounds.stream_state;
01104
01105     int str_size = sizeof(PATH_SOUND_STREAMS) + strlen(file) + sizeof(EXTENSION_SOUND_STREAM) + 1;
01106     char *path = NULL;
01107
01108     // analisis de reproduccion y carga de stream "para que no explote todo"
01109     switch (*state)
01110     {
01111     case SOUND_STREAM_STATE_PLAY:
01112         // pausa
01113         al_set_audio_stream_playing(*pt, false);
01114
01115     case SOUND_STREAM_STATE_PAUSE:
01116         // desacople del mixer
01117         al_detach_audio_stream(*pt);
01118         // destruccion
01119         al_destroy_audio_stream(*pt);
01120
01121     case SOUND_STREAM_STATE_NO_INIT:
01122         // armado del path del archivo
01123         path = malloc(str_size);
01124         must_init(path, file);
01125         memset(path, 0, str_size);
01126         strcat(path, PATH_SOUND_STREAMS);
01127         strcat(path, file);
01128         strcat(path, EXTENSION_SOUND_STREAM);
01129
01130         // carga del stream
01131         *pt = al_load_audio_stream(path, 2, 2048);
01132         if (*pt == NULL)
01133             return false;
01134
01135         free(path);
01136
01137         // modo de reproduccion
01138         al_set_audio_stream_playmode(*pt, ALLEGRO_PLAYMODE_LOOP);
01139
01140         // ganancia
01141         al_set_audio_stream_gain(*pt, gain);
01142
01143         // pausa
01144         al_set_audio_stream_playing(*pt, false);
01145
01146         //"para que suene" (acople al mixer)
01147         al_attach_audio_stream_to_mixer(sounds.stream, al_get_default_mixer());
01148
01149         // actualiza el nombre del ultimo stream inicializado
01150         strcpy(last_init_stream, file);
01151
01152         *state = SOUND_STREAM_STATE_PAUSE;
01153         break;
01154
01155     default:
01156         break;
01157     }
01158
01159     return true;
01160 }
01161
01162 static bool init_sample(ALLEGRO_SAMPLE **sample, const char *file)
01163 {
01164     if (file == NULL)
01165         return false;
01166
01167     int str_size = sizeof(PATH_SOUND_SAMPLES) + strlen(file) + sizeof(EXTENSION_SOUND_SAMPLE) + 1;
01168     char *path = NULL;
01169
01170     path = malloc(str_size);
01171     must_init(path, file);
01172     memset(path, 0, str_size);
01173     strcat(path, PATH_SOUND_SAMPLES);
01174     strcat(path, file);
01175     strcat(path, EXTENSION_SOUND_SAMPLE);
01176
```

```
01177     *sample = al_load_sample(path);
01178
01179     free(path);
01180
01181     if (*sample == NULL)
01182         return false;
01183
01184     return true;
01185 }
01186
01187 static void rick_init(void)
01188 {
01189     rick = algif_load_animation("../res/gifs/rick.gif");
01190     allegro_set_rick_flag(false);
01191 }
01192
01193 static void rick_deinit()
01194 {
01195     algif_destroy_animation(rick);
01196 }
```

4.18 src/platform/pc/allegro_stuff.h File Reference

Header del modulo allegro_stuff. Estructuras, prototipos de funciones globales.

```
#include <allegro5/allegro5.h>
#include <allegro5/allegro_font.h>
#include <allegro5/allegro_ttf.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_audio.h>
#include <allegro5/allegro_acodec.h>
#include <stdio.h>
#include "entities.h"
#include "geometry.h"
```

Include dependency graph for allegro_stuff.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct sprites_menu_t
- struct sprites_t

  Estructura principal de spritesheets.

Macros

- #define FPS 60

Enumerations

- enum **KEY_STATES** { **KEY_RELEASED** , **KEY_JUST_PRESSED** , **KEY_PRESSED** }

Functions

- void must_init (bool test, const char ∗description)

  Inicializa "cosas", y sale del programa si hay problemas, avisando dónde estuvo.
- void allegro_inits (void)

  Inicializaciones de allegro. Si algo falla, lo notifica por consola y finaliza el programa.
- void allegro_deinits (void)

  Desinicializaciones de allegro.
- void allegro_reinit_display (void)

  Reinicializa el display de allegro.
- void allegro_deinit_display (void)

  Desinicializa el display.
- unsigned char allegro_get_last_key (void)

  Devuelve la ultima tecla presionada registrada.
- void allegro_set_last_key (unsigned char allegro_key_code)

  Setea una ultima tecla presionada.
- ALLEGRO_EVENT_TYPE allegro_wait_for_event (void)

  Espera a que ocurra un evento y lo devuelve.
- ALLEGRO_EVENT ∗ allegro_get_next_event (void)

  Devuelve el proximo evento de la cola, si es que existe. De no haber, devuele false.
- ALLEGRO_EVENT allegro_get_var_event (void)

  Devuelve el evento de allegro.
- bool allegro_get_var_done (void)

  Devuelve flag de finalización del programa.
- bool allegro_get_var_redraw (void)

  Devuelve flag de renderización.
- void allegro_set_var_done (bool state)

  Setea flag de finalización del programa.
- void allegro_set_var_redraw (bool state)

  Setea flag de renderizacion.
- ALLEGRO_FONT ∗ allegro_get_var_font (void)

  Devuelve la fuente de allegro.
- int allegro_get_var_font_h (void)

  Devuelve el alto de un caracter de la funete usada.
- int allegro_get_var_font_w (void)

  Devuelve ancho de un caracter de la fuente usada.
- void allegro_clear_display (void)

  Pone negro el display.
- void allegro_draw_background (void)

  Dibuja la imagen de fondo.
- void allegro_draw_menu_background (int window)

  Dibuja imagen de fondo de un menu dado (sin highlight en ninguna opcion)

- bool allegro_is_event_queueVacia (void)

  Informa si al cola de eventos está vacía o no.
- ALLEGRO_EVENT_QUEUE * allegro_get_event_queue (void)

  Devuelve puntero a la cola de eventos.
- void allegro_set_var_event (ALLEGRO_EVENT event)

  Carga un evento de allegro.
- void allegro_sound_set_stream_credits (void)

  Selecciona musica de creditos. Comienza pausada.
- void allegro_sound_set_stream_main_menu (void)

  Selecciona musica de menu. Comienza pausada.
- void allegro_sound_set_stream_pause_menu (void)

  Selecciona musica de pausa. Comienza pausada.
- void allegro_sound_set_stream_ranking (void)

  Selecciona musica de ranking. Comienza pausada.
- void allegro_sound_set_stream_playing (void)

  Selecciona musica de jugando. Comienza pausada.
- void allegro_sound_set_stream_rick (void)
- void allegro_sound_set_stream_game_over (void)

  Selecciona musica de game over. Comienza pausada.
- void allegro_sound_toggle_stream (void)

  Cambia estado de reproduccion de la musica actual, si hay alguna seleccionada.
- void allegro_sound_play_stream (void)

  Reproduce la musica actual, si hay alguna seleccionada.
- void allegro_sound_pause_stream (void)

  Pausa la musica actual, si hay alguna seleccionada.
- void allegro_sound_restart_stream (void)

  Reinicia la musica actual, si hay alguna seleccionada.
- void allegro_sound_set_stream_gain_up (void)

  Aumenta en 0.1 la ganancia de stream.
- void allegro_sound_set_stream_gain_down (void)

  Reduce en 0.1 la ganancia de stream.
- void allegro_sound_play_effect_bonus (void)

  Reproduce efecto de bonus.
- void allegro_sound_play_effect_click (void)

  Reproduce efecto de click (seleccion de menu//aceptar//etc.)
- void allegro_sound_play_effect_crash (void)

  Reproduce efecto de choque.
- void allegro_sound_play_effect_drowned (void)

  Reproduce efecto de ahogado (toco agua)
- void allegro_sound_play_effect_goal (void)

  Reproduce efecto de 'llego a la meta'.
- void allegro_sound_play_effect_jump (void)

  Reproduce efecto de salto.
- void allegro_sound_play_effect_low_time (void)

  Reproduce efecto de 'queda poco tiempo'.
- void allegro_sound_play_effect_run_completed (void)

  Reproduce efecto de 'run completada' (llego 5 veces a la meta)
- void allegro_sound_play_effect_menu_enter (void)

  Reproduce efecto de 'menu enter'.
- void allegro_sound_play_effect_new_max_score (void)

  Reproduce efecto de 'new_max_score'.

- void allegro_sound_play_effect_exiting (void)

  Reproduce efecto de 'saliendo'.
- void allegro_sound_play_effect_no_time (void)

  Reproduce efecto de 'sin tiempo'.
- void allegro_sound_play_effect_coin_drop (void)

  Reproduce efecto `de moneda tirada`
- void allegro_draw_hitbox (int x, int y, int w, int h)

  Dibuja un contorno rectangular.
- void allegro_rick_on (void)
- bool allegro_get_rick_flag (void)
- void allegro_set_rick_flag (bool state)
- void allegro_rick_off (void)
- void allegro_rick_draw (void)

Variables

- sprites_t sprites

4.18.1 Detailed Description

Header del modulo allegro_stuff. Estructuras, prototipos de funciones globales.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file allegro_stuff.h.

4.18.2 Macro Definition Documentation

4.18.2.1 FPS `#define FPS 60`

Definition at line 37 of file allegro_stuff.h.

4.18.3 Enumeration Type Documentation

4.18.3.1 KEY_STATES `enum KEY_STATES`

Definition at line 43 of file allegro_stuff.h.

4.18.4 Function Documentation

4.18.4.1 allegro_clear_display() `void allegro_clear_display (`

```
void )
```

Pone negro el display.

Definition at line 458 of file allegro_stuff.c.

4.18.4.2 allegro_deinit_display() `void allegro_deinit_display (`

```
void )
```

Desinicializa el display.

Definition at line 382 of file allegro_stuff.c.

4.18.4.3 allegro_deinits() `void allegro_deinits (`

```
void )
```

Desinicializaciones de allegro.

Definition at line 336 of file allegro_stuff.c.

4.18.4.4 allegro_draw_background() `void allegro_draw_background (`

```
void )
```

Dibuja la imagen de fondo.

Definition at line 463 of file allegro_stuff.c.

4.18.4.5 allegro_draw_hitbox() `void allegro_draw_hitbox (`

```
int x,
int y,
int w,
int h )
```

Dibuja un contorno rectangular.

Parameters

x	Topleft x
y	Topleft y
w	Ancho
h	Largo

Definition at line 738 of file allegro_stuff.c.

4.18.4.6 allegro_draw_menu_background() `void allegro_draw_menu_background ( int window )`

Dibuja imagen de fondo de un menu dado (sin highlight en ninguna opcion)

Parameters

window	enum MENU_WINDOWS

Definition at line 468 of file allegro_stuff.c.

4.18.4.7 allegro_get_event_queue() `ALLEGRO_EVENT_QUEUE * allegro_get_event_queue ( void )`

Devuelve puntero a la cola de eventos.

Definition at line 478 of file allegro_stuff.c.

4.18.4.8 allegro_get_last_key() `unsigned char allegro_get_last_key ( void )`

Devuelve la ultima tecla presionada registrada.

Returns

unsigned char ALLEGRO_KEY_CODE

Definition at line 391 of file allegro_stuff.c.

4.18.4.9 allegro_get_next_event() `ALLEGRO_EVENT * allegro_get_next_event ( void )`

Devuelve el proximo evento de la cola, si es que existe. De no haber, devuele false.

Returns

ALLEGRO_EVENT*

Definition at line 408 of file allegro_stuff.c.

4.18.4.10 allegro_get_rick_flag() `bool allegro_get_rick_flag ( void )`

Definition at line 756 of file allegro_stuff.c.

4.18.4.11 allegro_get_var_done() `bool allegro_get_var_done ( void )`

Devuelve flag de finalización del programa.

Returns

true Finaliza

false No finaliza

Definition at line 423 of file allegro_stuff.c.

4.18.4.12 allegro_get_var_event() `ALLEGRO_EVENT allegro_get_var_event ( void )`

Devuelve el evento de allegro.

Returns

ALLEGRO_EVENT

Definition at line 418 of file allegro_stuff.c.

4.18.4.13 allegro_get_var_font() `ALLEGRO_FONT * allegro_get_var_font ( void )`

Devuelve la fuente de allegro.

Returns

ALLEGRO_FONT

Definition at line 443 of file allegro_stuff.c.

4.18.4.14 allegro_get_var_font_h() `int allegro_get_var_font_h ( void )`

Devuelve el alto de un caracter de la funete usada.

Returns

int alto

Definition at line 448 of file allegro_stuff.c.

4.18.4.15 allegro_get_var_font_w() `int allegro_get_var_font_w ( void )`

Devuelve ancho de un caracter de la fuente usada.

Returns

int ancho

Definition at line 453 of file allegro_stuff.c.

4.18.4.16 allegro_get_var_redraw() `bool allegro_get_var_redraw ( void )`

Devuelve flag de renderización.

Returns

true Renderiza

false No renderiza

Definition at line 428 of file allegro_stuff.c.

4.18.4.17 allegro_inits() `void allegro_inits ( void )`

Inicializaciones de allegro. Si algo falla, lo notifica por consola y finaliza el programa.

Definition at line 285 of file allegro_stuff.c.

4.18.4.18 allegro_is_event_queueVacia() `bool allegro_is_event_queueVacia ( void )`

Informa si al cola de eventos está vacía o no.

Returns

true Vacía

false No vacía

Definition at line 473 of file allegro_stuff.c.

4.18.4.19 allegro_reinit_display() `void allegro_reinit_display ( void )`

Reinicializa el display de allegro.

Definition at line 349 of file allegro_stuff.c.

4.18.4.20 allegro_rick_draw() `void allegro_rick_draw ( void )`

Definition at line 773 of file allegro_stuff.c.

4.18.4.21 allegro_rick_off() `void allegro_rick_off ( void )`

Definition at line 766 of file allegro_stuff.c.

4.18.4.22 allegro_rick_on() `void allegro_rick_on ( void )`

Definition at line 748 of file allegro_stuff.c.

4.18.4.23 allegro_set_last_key() `void allegro_set_last_key ( unsigned char allegro_key_code )`

Setea una ultima tecla presionada.

Parameters

allegro_key_code	ALLEGRO_KEY_CODE

Definition at line 396 of file allegro_stuff.c.

4.18.4.24 allegro_set_rick_flag() `void allegro_set_rick_flag ( bool state )`

Parameters

state	

Definition at line 761 of file allegro_stuff.c.

4.18.4.25 allegro_set_var_done() `void allegro_set_var_done ( bool state )`

Setea flag de finalización del programa.

Parameters

state	true or false

Definition at line 433 of file allegro_stuff.c.

4.18.4.26 allegro_set_var_event() `void allegro_set_var_event ( ALLEGRO_EVENT event )`

Carga un evento de allegro.

Parameters

event	

Definition at line 483 of file allegro_stuff.c.

4.18.4.27 allegro_set_var_redraw() `void allegro_set_var_redraw ( bool state )`

Setea flag de renderizacion.

Parameters

state	true or false

Definition at line 438 of file allegro_stuff.c.

4.18.4.28 allegro_sound_pause_stream() `void allegro_sound_pause_stream ( void )`

Pausa la musica actual, si hay alguna seleccionada.

Definition at line 631 of file allegro_stuff.c.

4.18.4.29 allegro_sound_play_effect_bonus() `void allegro_sound_play_effect_bonus ( void )`

Reproduce efecto de bonus.

Definition at line 669 of file allegro_stuff.c.

4.18.4.30 allegro_sound_play_effect_click() `void allegro_sound_play_effect_click ( void )`

Reproduce efecto de click (seleccion de menu//aceptar//etc.)

Definition at line 674 of file allegro_stuff.c.

4.18.4.31 allegro_sound_play_effect_coin_drop() `void allegro_sound_play_effect_coin_drop ( void )`

Reproduce efecto `de moneda tirada`

Definition at line 729 of file allegro_stuff.c.

4.18.4.32 allegro_sound_play_effect_crash() `void allegro_sound_play_effect_crash ( void )`

Reproduce efecto de choque.

Definition at line 679 of file allegro_stuff.c.

4.18.4.33 allegro_sound_play_effect_drowned() `void allegro_sound_play_effect_drowned ( void )`

Reproduce efecto de ahogado (toco agua)

Definition at line 684 of file allegro_stuff.c.

4.18.4.34 allegro_sound_play_effect_exiting() `void allegro_sound_play_effect_exiting ( void )`

Reproduce efecto de 'saliendo'.

Definition at line 719 of file allegro_stuff.c.

4.18.4.35 allegro_sound_play_effect_goal() `void allegro_sound_play_effect_goal ( void )`

Reproduce efecto de 'llego a la meta'.

Definition at line 689 of file allegro_stuff.c.

4.18.4.36 allegro_sound_play_effect_jump() `void allegro_sound_play_effect_jump ( void )`

Reproduce efecto de salto.

Definition at line 694 of file allegro_stuff.c.

4.18.4.37 allegro_sound_play_effect_low_time() `void allegro_sound_play_effect_low_time ( void )`

Reproduce efecto de 'queda poco tiempo'.

Definition at line 699 of file allegro_stuff.c.

4.18.4.38 allegro_sound_play_effect_menu_enter() `void allegro_sound_play_effect_menu_enter ( void )`

Reproduce efecto de 'menu enter'.

Definition at line 709 of file allegro_stuff.c.

4.18.4.39 allegro_sound_play_effect_new_max_score()

```
void allegro_sound_play_effect_new_max_score (
            void  )
```

Reproduce efecto de 'new_max_score'.

Definition at line 714 of file allegro_stuff.c.

4.18.4.40 allegro_sound_play_effect_no_time()

```
void allegro_sound_play_effect_no_time (
            void  )
```

Reproduce efecto de 'sin tiempo'.

Definition at line 724 of file allegro_stuff.c.

4.18.4.41 allegro_sound_play_effect_run_completed()

```
void allegro_sound_play_effect_run_completed (
            void  )
```

Reproduce efecto de 'run completada' (llego 5 veces a la meta)

Definition at line 704 of file allegro_stuff.c.

4.18.4.42 allegro_sound_play_stream()

```
void allegro_sound_play_stream (
            void  )
```

Reproduce la musica actual, si hay alguna seleccionada.

Definition at line 622 of file allegro_stuff.c.

4.18.4.43 allegro_sound_restart_stream()

```
void allegro_sound_restart_stream (
            void  )
```

Reinicia la musica actual, si hay alguna seleccionada.

Definition at line 640 of file allegro_stuff.c.

4.18.4.44 allegro_sound_set_stream_credits()

```
void allegro_sound_set_stream_credits (
            void  )
```

Selecciona musica de creditos. Comienza pausada.

Definition at line 491 of file allegro_stuff.c.

4.18.4.45 allegro_sound_set_stream_gain_down() `void allegro_sound_set_stream_gain_down ( void )`

Reduce en 0.1 la ganancia de stream.

Definition at line 657 of file allegro_stuff.c.

4.18.4.46 allegro_sound_set_stream_gain_up() `void allegro_sound_set_stream_gain_up ( void )`

Aumenta en 0.1 la ganancia de stream.

Definition at line 648 of file allegro_stuff.c.

4.18.4.47 allegro_sound_set_stream_game_over() `void allegro_sound_set_stream_game_over ( void )`

Selecciona musica de game over. Comienza pausada.

Definition at line 587 of file allegro_stuff.c.

4.18.4.48 allegro_sound_set_stream_main_menu() `void allegro_sound_set_stream_main_menu ( void )`

Selecciona musica de menu. Comienza pausada.

Definition at line 507 of file allegro_stuff.c.

4.18.4.49 allegro_sound_set_stream_pause_menu() `void allegro_sound_set_stream_pause_menu ( void )`

Selecciona musica de pausa. Comienza pausada.

Definition at line 523 of file allegro_stuff.c.

4.18.4.50 allegro_sound_set_stream_playing() `void allegro_sound_set_stream_playing ( void )`

Selecciona musica de jugando. Comienza pausada.

Definition at line 555 of file allegro_stuff.c.

4.18.4.51 allegro_sound_set_stream_ranking() `void allegro_sound_set_stream_ranking ( void )`

Selecciona musica de ranking. Comienza pausada.

Definition at line 539 of file allegro_stuff.c.

4.18.4.52 allegro_sound_set_stream_rick() `void allegro_sound_set_stream_rick ( void )`

Definition at line 571 of file allegro_stuff.c.

4.18.4.53 allegro_sound_toggle_stream() `void allegro_sound_toggle_stream ( void )`

Cambia estado de reproduccion de la musica actual, si hay alguna seleccionada.

Definition at line 606 of file allegro_stuff.c.

4.18.4.54 allegro_wait_for_event() `ALLEGRO_EVENT_TYPE allegro_wait_for_event ( void )`

Espera a que ocurra un evento y lo devuelve.

Returns

ALLEGRO_EVENT_TYPE

Definition at line 401 of file allegro_stuff.c.

4.18.4.55 must_init() `void must_init ( bool test, const char * description )`

Inicializa "cosas", y sale del programa si hay problemas, avisando dónde estuvo.

Parameters

test	Handler//booleano con status de la inicialización.
description	String con la descripción/nombre de la "cosa" a inicializar.

Definition at line 275 of file allegro_stuff.c.

4.18.5 Variable Documentation

4.18.5.1 sprites `sprites_t sprites [extern]`

Definition at line 151 of file allegro_stuff.c.

4.19 allegro_stuff.h

Go to the documentation of this file.

```
00001
00012 #ifndef _ALLEGRO_STUFF_H_
00013 #define _ALLEGRO_STUFF_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include <allegro5/allegro5.h>
00020 #include <allegro5/allegro_font.h>
00021 #include <allegro5/allegro_ttf.h>
00022 #include <allegro5/allegro_primitives.h>
00023 #include <allegro5/allegro_image.h>
00024
00025 #include <allegro5/allegro_audio.h>
00026 #include <allegro5/allegro_acodec.h>
00027
00028 #include <stdio.h>
00029
00030 #include "entities.h"
00031 #include "geometry.h"
00032
00033 /*****************************************************************************
00034  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00035  *****************************************************************************/
00036
00037 #define FPS 60
00038
00039 /*****************************************************************************
00040  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00041  *****************************************************************************/
00042
00043 enum KEY_STATES
00044 {
00045     KEY_RELEASED,
00046     KEY_JUST_PRESSED,
00047     KEY_PRESSED
00048 };
00049
00050 typedef struct
00051 {
00052
00053 } sprites_menu_t;
00054
00059 typedef struct
00060 {
00061     ALLEGRO_BITMAP *frog_uncut;
00062     ALLEGRO_BITMAP *frog[8];
00063
00064     ALLEGRO_BITMAP *background;
00065
00066     ALLEGRO_BITMAP *log;
00067
00068     ALLEGRO_BITMAP *cars_uncut;
00069     ALLEGRO_BITMAP *car[CAR_TYPE_N];
00070
00071     ALLEGRO_BITMAP *turtle_uncut;
00072     ALLEGRO_BITMAP *turtle[TURTLE_FRAMES];
00073
00074     ALLEGRO_BITMAP *heart;
00075
00076     struct
00077     {
00078         ALLEGRO_BITMAP *uncut;
00079         ALLEGRO_BITMAP *option[MENU_STATE_MAX];
00080         ALLEGRO_BITMAP *background;
```

```
00081     } menu[MENU_WINDOW_MAX];
00082
00083     ALLEGRO_BITMAP *credits;
00084
00085     ALLEGRO_BITMAP *name;
00086
00087     ALLEGRO_BITMAP *icon;
00088
00089     ALLEGRO_BITMAP *dead;
00090
00091     struct
00092     {
00093         ALLEGRO_BITMAP *uncut;
00094         ALLEGRO_BITMAP *frame[SPRITE_COIN_FRAMES];
00095     } coin;
00096
00097     struct
00098     {
00099         ALLEGRO_BITMAP *uncut;
00100         ALLEGRO_BITMAP *frame[SPRITE_SPLASH_FRAMES];
00101     } splash;
00102
00103     ALLEGRO_BITMAP *border;
00104
00105 } sprites_t;
00106
00107 /*****************************************************************************
00108  * VARIABLE PROTOTYPES WITH GLOBAL SCOPE
00109  *****************************************************************************/
00110
00111 // estructura con punteros a sprites
00112 extern sprites_t sprites;
00113
00114 /*****************************************************************************
00115  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00116  *****************************************************************************/
00117
00124 void must_init(bool test, const char *description);
00125
00130 void allegro_inits(void);
00131
00136 void allegro_deinits(void);
00137
00142 void allegro_reinit_display(void);
00143
00148 void allegro_deinit_display(void);
00149
00155 unsigned char allegro_get_last_key(void);
00156
00162 void allegro_set_last_key(unsigned char allegro_key_code);
00163
00169 ALLEGRO_EVENT_TYPE allegro_wait_for_event(void);
00170
00176 ALLEGRO_EVENT *allegro_get_next_event(void);
00177
00183 ALLEGRO_EVENT allegro_get_var_event(void);
00184
00191 bool allegro_get_var_done(void);
00192
00199 bool allegro_get_var_redraw(void);
00200
00206 void allegro_set_var_done(bool state);
00207
00213 void allegro_set_var_redraw(bool state);
00214
00220 ALLEGRO_FONT *allegro_get_var_font(void);
00221
00227 int allegro_get_var_font_h(void);
00228
00234 int allegro_get_var_font_w(void);
00235
00240 void allegro_clear_display(void);
00241
00246 void allegro_draw_background(void);
00247
00253 void allegro_draw_menu_background(int window);
00254
00261 bool allegro_is_event_queueVacia(void);
00262
00267 ALLEGRO_EVENT_QUEUE *allegro_get_event_queue(void);
00268
00274 void allegro_set_var_event(ALLEGRO_EVENT event);
00275
00276 #pragma region allegro_sound
00277
00278 #pragma region allegro_sound_set_stream
00283 void allegro_sound_set_stream_credits(void);
```

```
00284
00289 void allegro_sound_set_stream_main_menu(void);
00290
00295 void allegro_sound_set_stream_pause_menu(void);
00296
00301 void allegro_sound_set_stream_ranking(void);
00302
00307 void allegro_sound_set_stream_playing(void);
00308
00313 void allegro_sound_set_stream_rick(void);
00314
00319 void allegro_sound_set_stream_game_over(void);
00320
00321 #pragma endregion allegro_sound_set_stream
00322
00323 #pragma region allegro_sound_control
00328 void allegro_sound_toggle_stream(void);
00329
00334 void allegro_sound_play_stream(void);
00335
00340 void allegro_sound_pause_stream(void);
00341
00346 void allegro_sound_restart_stream(void);
00347
00352 void allegro_sound_set_stream_gain_up(void);
00353
00358 void allegro_sound_set_stream_gain_down(void);
00359
00360 #pragma endregion allegro_sound_control
00361
00362 #pragma region allegro_sound_play_sample
00367 void allegro_sound_play_effect_bonus(void);
00368
00373 void allegro_sound_play_effect_click(void);
00374
00379 void allegro_sound_play_effect_crash(void);
00380
00385 void allegro_sound_play_effect_drowned(void);
00386
00391 void allegro_sound_play_effect_goal(void);
00392
00397 void allegro_sound_play_effect_jump(void);
00398
00403 void allegro_sound_play_effect_low_time(void);
00404
00409 void allegro_sound_play_effect_run_completed(void);
00410
00415 void allegro_sound_play_effect_menu_enter(void);
00416
00421 void allegro_sound_play_effect_new_max_score(void);
00422
00427 void allegro_sound_play_effect_exiting(void);
00428
00433 void allegro_sound_play_effect_no_time(void);
00434
00439 void allegro_sound_play_effect_coin_drop(void);
00440
00441 #pragma endregion allegro_sound_play_sample
00442
00443 #pragma endregion allegro_sound
00444
00453 void allegro_draw_hitbox(int x, int y, int w, int h);
00454
00459 void allegro_rick_on(void);
00460
00465 bool allegro_get_rick_flag(void);
00466
00472 void allegro_set_rick_flag(bool state);
00473
00478 void allegro_rick_off(void);
00479
00484 void allegro_rick_draw(void);
00485
00486 /*****************************************************************************
00487  *****************************************************************************/
00488
00489 #endif // _ALLEGRO_STUFF_H_
```

4.20 src/platform/pc/display.c File Reference

Source del modulo display, orientado a PC. Rutinas relativas a la visualización del juego en pantalla, llamadas por la FSM.

```
#include "../../display.h"
#include "../../ranking.h"
#include "allegro_stuff.h"
#include "game_data.h"
#include <pthread.h>
```

Include dependency graph for display.c:

Macros

- #define CREDITS_SCROLL_SPEED 1
- #define RANKING_PLAYER_X 90
- #define RANKING_SCORE_X 500
- #define RANKING_START_Y 100

Functions

- bool iniciarDisplay ()

  Inicializa el display de la plataforma.
- void actualizarDisplay ()

  Actualiza el display de la plataforma.
- void limpiarDisplay ()

  Limpia el display de la plataforma.
- void mostrarTexto (char ∗txt, int pos)

  Muestra un texto dado en una posicion dada (retiene el flujo)
- void dejarTexto (char ∗txt, int pos, bool repetir)

  Deja el texto en la posición data (no retiene)
- void cargarRanking (void)

  Inicia muestreo de ranking en la plataforma.
- void mostrarRanking (void)

  Bucle que muestra el ranking. Finaliza desde dentro, y hace que finalice el thread de ranking.
- void cargarCreditos (void)

  Inicializa los cretidos en la plataforma.
- void mostrarCreditos (void)

  Bucle que muestra los creditos. Finaliza desde dentro, y hace que finalice el thread de ranking.
- void reconfigurarDisplayON (void)

  Reconfigura el display de la plataforma y lo habilita.
- void reconfigurarDisplayOFF (void)

  Reconfigura el display de la plataforma y lo deshabilita.

4.20.1 Detailed Description

Source del modulo display, orientado a PC. Rutinas relativas a la visualización del juego en pantalla, llamadas por la FSM.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file display.c.

4.20.2 Macro Definition Documentation

4.20.2.1 CREDITS_SCROLL_SPEED `#define CREDITS_SCROLL_SPEED 1`

Definition at line 28 of file display.c.

4.20.2.2 RANKING_PLAYER_X `#define RANKING_PLAYER_X 90`

Definition at line 30 of file display.c.

4.20.2.3 RANKING_SCORE_X `#define RANKING_SCORE_X 500`

Definition at line 31 of file display.c.

4.20.2.4 RANKING_START_Y `#define RANKING_START_Y 100`

Definition at line 32 of file display.c.

4.20.3 Function Documentation

4.20.3.1 actualizarDisplay() `void actualizarDisplay ( )`

Actualiza el display de la plataforma.

Definition at line 58 of file display.c.

4.20.3.2 cargarCreditos() `void cargarCreditos ( void )`

Inicializa los cretidos en la plataforma.

Definition at line 135 of file display.c.

4.20.3.3 cargarRanking() `void cargarRanking ( void )`

Inicia muestreo de ranking en la plataforma.

Parameters

txt	

Definition at line 74 of file display.c.

4.20.3.4 dejarTexto() `void dejarTexto ( char * txt, int pos, bool repetir )`

Deja el texto en la posición data (no retiene)

Parameters

txt	
pos	
repetir	

Definition at line 70 of file display.c.

4.20.3.5 iniciarDisplay() `bool iniciarDisplay ( )`

Inicializa el display de la plataforma.

Returns

true Exito

false Error

Definition at line 48 of file display.c.

4.20.3.6 limpiarDisplay() `void limpiarDisplay ( )`

Limpia el display de la plataforma.

Definition at line 62 of file display.c.

4.20.3.7 mostrarCreditos() `void mostrarCreditos ( void )`

Bucle que muestra los creditos. Finaliza desde dentro, y hace que finalice el thread de ranking.

Returns

true No finaliz

false Finaliza

Definition at line 140 of file display.c.

4.20.3.8 mostrarRanking() `void mostrarRanking ( void )`

Bucle que muestra el ranking. Finaliza desde dentro, y hace que finalice el thread de ranking.

Definition at line 131 of file display.c.

4.20.3.9 mostrarTexto() `void mostrarTexto ( char * txt, int pos )`

Muestra un texto dado en una posicion dada (retiene el flujo)

Parameters

txt	Texto
pos	Posicion

Definition at line 66 of file display.c.

4.20.3.10 reconfigurarDisplayOFF() void reconfigurarDisplayOFF (void)

Reconfigura el display de la plataforma y lo deshabilita.

Definition at line 162 of file display.c.

4.20.3.11 reconfigurarDisplayON() void reconfigurarDisplayON (void)

Reconfigura el display de la plataforma y lo habilita.

Definition at line 157 of file display.c.

4.21 display.c

Go to the documentation of this file.

```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "../../display.h"
00017 #include "../../ranking.h"
00018
00019 #include "allegro_stuff.h"
00020 #include "game_data.h"
00021
00022 #include <pthread.h>
00023
00024 /*******************************************************************************
00025  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00026  ******************************************************************************/
00027
00028 #define CREDITS_SCROLL_SPEED 1
00029
00030 #define RANKING_PLAYER_X 90
00031 #define RANKING_SCORE_X 500
00032 #define RANKING_START_Y 100
00033
00034 /*******************************************************************************
00035  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00036  ******************************************************************************/
00037
00038 static pthread_mutex_t lock;
00039
00040 static int credits_scroll_cont;
00041
00042 /*******************************************************************************
00043  *******************************************************************************
00044                         GLOBAL FUNCTION DEFINITIONS
00045  *******************************************************************************
00046  ******************************************************************************/
00047
00048 bool iniciarDisplay()
00049 {
00050     if (pthread_mutex_init(&lock, NULL) != 0)
00051         return 1;
00052
00053     allegro_inits();
00054
00055     return 0;
00056 }
00057
```

```
00058 void actualizarDisplay()
00059 {
00060 }
00061
00062 void limpiarDisplay()
00063 {
00064 }
00065
00066 void mostrarTexto(char *txt, int pos)
00067 {
00068 }
00069
00070 void dejarTexto(char *txt, int pos, bool repetir)
00071 {
00072 }
00073
00074 void cargarRanking(void)
00075 {
00076     int i = 0;
00077
00078     int lines = getRankingLineas();
00079     char **names = getRankingNombres();
00080     unsigned long long *scores = getRankingPuntos();
00081
00082     allegro_clear_display();
00083
00084     al_draw_text(allegro_get_var_font(),
00085                  al_map_rgb(10, 180, 10),
00086                  RANKING_PLAYER_X,
00087                  60,
00088                  0,
00089                  "Jugador");
00090
00091     al_draw_text(allegro_get_var_font(),
00092                  al_map_rgb(10, 180, 10),
00093                  RANKING_SCORE_X,
00094                  60,
00095                  0,
00096                  "Puntaje");
00097
00098     if (lines)
00099     {
00100         for (i = 0; i < lines; i++)
00101         {
00102             al_draw_textf(allegro_get_var_font(),
00103                           al_map_rgb(255, 255, 255),
00104                           RANKING_PLAYER_X,
00105                           RANKING_START_Y + i * 20,
00106                           0,
00107                           "%s", names[i]);
00108
00109             al_draw_textf(allegro_get_var_font(),
00110                           al_map_rgb(255, 255, 255),
00111                           RANKING_SCORE_X,
00112                           RANKING_START_Y + i * 20,
00113                           0,
00114                           "%lld", scores[i]);
00115         }
00116     }
00117
00118     else
00119     {
00120         al_draw_text(allegro_get_var_font(),
00121                      al_map_rgb(255, 255, 255),
00122                      DISPLAY_W / 2 - al_get_text_width(allegro_get_var_font(), "Ningún jugador
       registrado") / 2,
00123                      RANKING_START_Y,
00124                      0,
00125                      "Ningún jugador registrado");
00126     }
00127
00128     al_flip_display();
00129 }
00130
00131 void mostrarRanking(void)
00132 {
00133 }
00134
00135 void cargarCreditos(void)
00136 {
00137     credits_scroll_cont = 0;
00138 }
00139
00140 void mostrarCreditos(void)
00141 {
00142     if (allegro_get_var_redraw())
00143     {
```

```
00144
00145         credits_scroll_cont -= CREDITS_SCROLL_SPEED;
00146         if (credits_scroll_cont == -CREDITS_SCREEN_FINAL)
00147             credits_scroll_cont = CREDITS_SCREEN_START;
00148
00149         allegro_clear_display();
00150         al_draw_bitmap(sprites.credits, 0, credits_scroll_cont, 0);
00151         al_flip_display();
00152
00153         allegro_set_var_redraw(false);
00154     }
00155 }
00156
00157 void reconfigurarDisplayON(void)
00158 {
00159     allegro_reinit_display();
00160 }
00161
00162 void reconfigurarDisplayOFF(void)
00163 {
00164     allegro_deinit_display();
00165 }
```

4.22 display.c

```
00001
00008 /*******************************************************************************
00009  * INCLUDE HEADER FILES
00010  ******************************************************************************/
00011
00012 #include "../../display.h"
00013 #include "../../ranking.h"
00014
00015 #include "mensajes.h"
00016 #include "bitmap.h"
00017 #include "disdrv.h"
00018
00019 #include <unistd.h>
00020 #include <pthread.h>
00021 #include <string.h>
00022
00023 /*******************************************************************************
00024  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00025  ******************************************************************************/
00026
00027 #define CASTEAR_POSICION(pos) ((pos) == POS_CREDITOS ? POS_MSJ3 : (((pos) == POS_OPCION) || ((pos) ==
       POS_RANKING_2) ? POS_MSJ2 : POS_MSJ1))
00028
00029 /*******************************************************************************
00030  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00031  ******************************************************************************/
00032
00033 /*******************************************************************************
00034  * VARIABLES WITH GLOBAL SCOPE
00035  ******************************************************************************/
00036
00037 matriz_t disp_matriz;
00038
00039 /*******************************************************************************
00040  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00041  ******************************************************************************/
00042
00049 static void *threadTextoDisplay(void *ptr);
00050 static void *threadPresentacion(void *ptr);
00051 static void ulltoa(unsigned long long num, char *str);
00052
00053 /*******************************************************************************
00054  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00055  ******************************************************************************/
00056
00057 static const clock_t TIEMPO_SLEEP_DISPLAY = CLOCKS_PER_SEC » 3;
00058
00059 /*******************************************************************************
00060  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00061  ******************************************************************************/
00062
00063 static pthread_mutex_t lock;
00064 static pthread_t ttextodisplay, tpresentacion;
00065 static mensaje_t texto1, texto2, texto3;
00066 static bool thread_encendido, thread_presentacion_encendido;
00067
00068 static int lines, i;
00069 static char **names;
00070 static unsigned long long *scores;
```

```
00071
00072 static char *creditos_cadenas[] = {"PROGRAMACION TPF 2021 1C", "FROGGER", "AUTORES", "ALEJANDRO HEIR",
       "FRANCO AGGRIPINO", "MATIAS ALVAREZ", "TOMAS CASTRO"};
00073
00074 /*******************************************************************************
00075  *******************************************************************************
00076                         GLOBAL FUNCTION DEFINITIONS
00077  *******************************************************************************
00078  ******************************************************************************/
00079
00080 bool iniciarDisplay()
00081 {
00082     if (pthread_mutex_init(&lock, NULL) != 0)
00083     {
00084         return 1;
00085     }
00086
00087     disp_init();  // inicializa el display
00088     disp_clear(); // limpia todo el display
00089
00090     return 0;
00091 }
00092
00093 void actualizarDisplay()
00094 {
00095     pthread_mutex_lock(&lock);
00096     for (int i = DISP_MIN; i <= (DISP_MAX_Y); i++)
00097         for (int j = DISP_MIN; j <= (DISP_MAX_X); j++)
00098             disp_write((dcoord_t){j, i}, disp_matriz[i] & (0x8000 >> j));
00099
00100     disp_update();
00101     pthread_mutex_unlock(&lock);
00102 }
00103
00104 void limpiarDisplay()
00105 {
00106     texto1.habilitacion = false;
00107     texto2.habilitacion = false;
00108     texto3.habilitacion = false;
00109
00110     if (thread_encendido)
00111     {
00112         thread_encendido = false;
00113         pthread_join(ttextodisplay, NULL);
00114     }
00115     if (thread_presentacion_encendido)
00116     {
00117         thread_presentacion_encendido = false;
00118         pthread_join(tpresentacion, NULL);
00119     }
00120
00121     limpiarMatriz(disp_matriz);
00122     actualizarDisplay();
00123 }
00124
00125 void mostrarTexto(char *txt, int pos)
00126 {
00127     int posicion = CASTEAR_POSICION(pos);
00128     mensaje_t msj = mensaje(txt, posicion, false);
00129     while (!renglonIzquierdoLibre(&msj))
00130     {
00131         usleep(TIEMPO_SLEEP_DISPLAY);
00132         moverMensaje(&msj);
00133         copiarMatrizRenglon(disp_matriz, msj.renglon, msj.posicion);
00134         actualizarDisplay();
00135     }
00136 }
00137
00138 void dejarTexto(char *txt, int pos, bool repetir)
00139 {
00140     int posicion = CASTEAR_POSICION(pos);
00141
00142     switch (posicion)
00143     {
00144     case POS_MSJ1:
00145         texto1 = mensaje(txt, posicion, repetir);
00146         break;
00147     case POS_MSJ2:
00148         texto2 = mensaje(txt, posicion, repetir);
00149         break;
00150     default:
00151         limpiarDisplay();
00152         texto3 = mensaje(txt, posicion, repetir);
00153     }
00154
00155     if (!thread_encendido)
00156     {
```

```
00157         thread_encendido = true;
00158         pthread_create(&ttextodisplay, NULL, threadTextoDisplay, NULL);
00159     }
00160 }
00161
00162 void cargarRanking(void)
00163 {
00164     borrarRenglon(texto2.renglon);
00165     lines = getRankingLineas();
00166     if (lines <= 0)
00167         dejarTexto("NINGUNA PARTIDA COMPLETADA AUN", POS_MSJ2, true);
00168     else
00169     {
00170         names = getRankingNombres();
00171         scores = getRankingPuntos();
00172         i = 0;
00173     }
00174 }
00175
00176 void mostrarRanking(void)
00177 {
00178     if (renglonIzquierdoLibre(&texto2) && lines > 0)
00179     { // si se acabó lo que tenía para mostrar abajo, busco la siguiente posición
00180         renglon_t r = {0};
00181         uintARenglon(i + 1, r);
00182         copiarMatrizRenglon(disp_matriz, r, POS_MSJ1); // se pone la posición en el renglón de arriba
00183         actualizarDisplay();
00184
00185         char score_str[L_MAX], puesto_msj[L_MAX];
00186
00187         strcpy(puesto_msj, names[i]);
00188         strcat(puesto_msj, " ");
00189         ulltoa(scores[i], score_str);
00190         strcat(puesto_msj, score_str);           // se arma un string con el nombre de jugador y la
      puntuación
00191         dejarTexto(puesto_msj, POS_MSJ2, false); // se muestra el string en la posición de abajo hasta
      que
00192
00193         if (++i >= lines) // apunto a la siguiente posición
00194             i = 0;
00195     }
00196 }
00197
00198 void cargarCreditos()
00199 {
00200     i = 0;
00201 }
00202
00203 void mostrarCreditos(void)
00204 {
00205     if ((renglonIzquierdoLibre(&texto3) && !thread_presentacion_encendido) && i < 7)
00206     {
00207         switch (i)
00208         {
00209         case 1:
00210             limpiarDisplay();
00211             thread_presentacion_encendido = true;
00212             pthread_create(&tpresentacion, NULL, threadPresentacion, NULL);
00213             break;
00214         case 2:
00215             pthread_join(tpresentacion, NULL);
00216         default:
00217             dejarTexto(creditos_cadenas[i], POS_CREDITOS, false);
00218         }
00219         i++;
00220     }
00221 }
00222
00223 void reconfigurarDisplayON(void)
00224 {
00225 }
00226
00227 void reconfigurarDisplayOFF(void)
00228 {
00229 }
00230
00231 /*******************************************************************************
00232  *******************************************************************************
00233                         LOCAL FUNCTION DEFINITIONS
00234  *******************************************************************************
00235  *******************************************************************************/
00236
00237 static void *threadTextoDisplay(void *ptr)
00238 {
00239     while (thread_encendido && (texto1.habilitacion || texto2.habilitacion || texto3.habilitacion))
00240     {
00241         usleep(TIEMPO_SLEEP_DISPLAY);
```

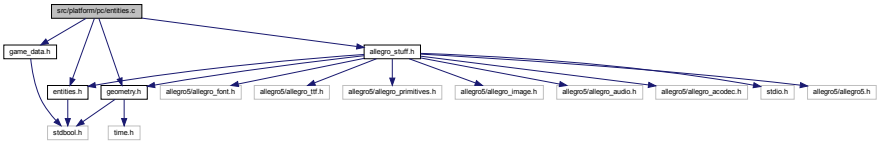
```
00242         if (texto1.habilitacion)
00243         {
00244             moverMensaje(&texto1);
00245             copiarMatrizRenglon(disp_matriz, texto1.renglon, POS_MSJ1);
00246         }
00247         if (texto2.habilitacion)
00248         {
00249             moverMensaje(&texto2);
00250             copiarMatrizRenglon(disp_matriz, texto2.renglon, POS_MSJ2);
00251         }
00252         if (texto3.habilitacion)
00253         {
00254             moverMensaje(&texto3);
00255             copiarMatrizRenglon(disp_matriz, texto3.renglon, POS_MSJ3);
00256         }
00257         actualizarDisplay();
00258     }
00259
00260     thread_encendido = false;
00261     pthread_exit(NULL);
00262 }
00263
00264 static void *threadPresentacion(void *ptr)
00265 {
00266     int coordenadas[][2] = {{0, 0}, {4, 1}, {8, 3}, {1, 8}, {5, 9}, {9, 10}, {13, 11}};
00267     int j;
00268     for (j = 0; j < 14 && thread_presentacion_encendido; j++)
00269     {
00270         matriz_t letra_matriz;
00271         charAMatriz(creditos_cadenas[1][j % 7], letra_matriz, coordenadas[j % 7]);
00272         if (j >= 7)
00273             matrizXor(disp_matriz, letra_matriz);
00274         else
00275             matrizOr(disp_matriz, letra_matriz);
00276         actualizarDisplay();
00277         usleep(TIEMPO_SLEEP_DISPLAY);
00278     }
00279
00280     thread_presentacion_encendido = false;
00281
00282     return NULL;
00283 }
00284
00285 static void ulltoa(unsigned long long num, char *str)
00286 {
00287     unsigned long long sum = num;
00288     int i = 0;
00289     int digit;
00290     do
00291     {
00292         digit = sum % 10;
00293         str[i++] = '0' + digit;
00294         sum /= 10;
00295     } while (sum);
00296     str[i--] = '\0';
00297
00298     int j = 0;
00299     char ch;
00300     while (i > j)
00301     {
00302         ch = str[i];
00303         str[i--] = str[j];
00304         str[j++] = ch;
00305     }
00306 }
```

4.23 src/platform/pc/entities.c File Reference

Source del modulo entities. Se encarga de la creacion, actualización y muestreo de las entitades implementadas en PC.

```
#include "entities.h"
#include "allegro_stuff.h"
#include "geometry.h"
#include "game_data.h"
```

Include dependency graph for entities.c:



Data Structures

- struct frog_t
- struct car_t
- struct log_t
- struct turtle_pack_t
- struct coin_t

Macros

- #define LOGS_SPAWN_MIN 1
- #define LOGS_SPAWN_MAX 3
- #define LOGS_SPAWN_FRAMES 60
- #define LOGS_BASE_SPEED 1
- #define LOGS_MAX_USED 7
- #define LOGS_EXTRA_SEPARATOR LOG_W / 2
- #define CARS_SPAWN_MIN 2
- #define CARS_SPAWN_FRAMES 60
- #define CARS_BASE_SPEED 1
- #define CARS_MAX_USED 15
- #define CAR_SPEED_INCREASE 2
- #define CAR_WAIT_INCREASE 1
- #define CARS_EXTRA_SEPARATOR CAR_W ∗ 2
- #define TURTLES_MIN_PER_PACK 1
- #define TURTLES_MAX_PER_PACK 2
- #define TURTLES_SPAWN_FRAMES 60
- #define TURTLES_SPAWN_MIN 1
- #define TURTLES_SPAWN_MAX 2
- #define TURTLES_MAX_USED 7
- #define TURTLES_BASE_SPEED 2
- #define TURTLES_FRAME_TIMEOUT_SURFACE 10
- #define TURTLES_FRAME_TIMEOUT_GOING_DOWN 50
- #define TURTLES_FRAME_TIMEOUT_WATER 20
- #define TURTLES_FRAME_TIMEOUT_GOING_UP 10
- #define TURTLES_SURFACE_FRAMES_MIN 80
- #define TURTLES_SURFACE_FRAMES_MAX 500
- #define TURTLES_WATER_FRAMES_MIN 60
- #define TURTLES_WATER_FRAMES_MAX 100
- #define TURTLES_EXTRA_SEPARATOR TURTLE_SIDE ∗ 2
- #define COIN_SPAWN_FRAMES_MIN 300
- #define COIN_SPAWN_FRAMES_MAX 600
- #define COIN_DESPAWN_FRAMES_MIN 700
- #define COIN_DESPAWN_FRAMES_MAX 900

- #define COIN_FRAME_RATE 10
- #define COIN_FRAMES_TO_WARN_A 250
- #define COIN_FRAMES_TO_WARN_B 100
- #define COIN_WARNING_FRAMES_A 20
- #define COIN_WARNING_FRAMES_B 10
- #define SPRITE_DEAD_TIMEOUT 80
- #define SPRITE_SPLASH_RATE 10

Enumerations

- enum **TURTLE_STATES** { **TURTLE_STATE_SURFACE** , **TURTLE_STATE_GOING_DOWN** , **TURTLE_STATE_WATER** , **TURTLE_STATE_GOING_UP** }
- enum **FROG_STATES** {
  FROG_STATE_ROAD , **FROG_STATE_WATER** , **FROG_STATE_LOG** , **FROG_STATE_TURTLE** ,
  FROG_STATE_GOAL , **FROG_STATE_GOAL_COIN** , **FROG_STATE_CRASH_CAR** , **FROG_STATE_CRASH_WALL** ,
  FROG_STATE_BOUNCING_WALL }

Functions

- void entities_init (void)

  Inicializa las entidades.
- void entities_update ()

  Actualiza las entidades.
- void entities_draw ()

  Dibuja las entidades.
- void entities_move_frog (unsigned char direction)

  Indica que la rana debe dar un salto en la direccion dada.

4.23.1 Detailed Description

Source del modulo entities. Se encarga de la creacion, actualización y muestreo de las entitades implementadas en PC.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file entities.c.

4.23.2 Macro Definition Documentation

4.23.2.1 CAR_SPEED_INCREASE `#define CAR_SPEED_INCREASE 2`

Definition at line 38 of file entities.c.

4.23.2.2 CAR_WAIT_INCREASE `#define CAR_WAIT_INCREASE 1`

Definition at line 39 of file entities.c.

4.23.2.3 CARS_BASE_SPEED `#define CARS_BASE_SPEED 1`

Definition at line 36 of file entities.c.

4.23.2.4 CARS_EXTRA_SEPARATOR `#define CARS_EXTRA_SEPARATOR CAR_W * 2`

Definition at line 40 of file entities.c.

4.23.2.5 CARS_MAX_USED `#define CARS_MAX_USED 15`

Definition at line 37 of file entities.c.

4.23.2.6 CARS_SPAWN_FRAMES `#define CARS_SPAWN_FRAMES 60`

Definition at line 35 of file entities.c.

4.23.2.7 CARS_SPAWN_MIN `#define CARS_SPAWN_MIN 2`

Definition at line 34 of file entities.c.

4.23.2.8 COIN_DESPAWN_FRAMES_MAX `#define COIN_DESPAWN_FRAMES_MAX 900`

Definition at line 62 of file entities.c.

4.23.2.9 COIN_DESPAWN_FRAMES_MIN `#define COIN_DESPAWN_FRAMES_MIN 700`

Definition at line 61 of file entities.c.

4.23.2.10 COIN_FRAME_RATE `#define COIN_FRAME_RATE 10`

Definition at line 63 of file entities.c.

4.23.2.11 COIN_FRAMES_TO_WARN_A `#define COIN_FRAMES_TO_WARN_A 250`

Definition at line 64 of file entities.c.

4.23.2.12 COIN_FRAMES_TO_WARN_B `#define COIN_FRAMES_TO_WARN_B 100`

Definition at line 65 of file entities.c.

4.23.2.13 COIN_SPAWN_FRAMES_MAX `#define COIN_SPAWN_FRAMES_MAX 600`

Definition at line 60 of file entities.c.

4.23.2.14 COIN_SPAWN_FRAMES_MIN `#define COIN_SPAWN_FRAMES_MIN 300`

Definition at line 59 of file entities.c.

4.23.2.15 COIN_WARNING_FRAMES_A `#define COIN_WARNING_FRAMES_A 20`

Definition at line 66 of file entities.c.

4.23.2.16 COIN_WARNING_FRAMES_B `#define COIN_WARNING_FRAMES_B 10`

Definition at line 67 of file entities.c.

4.23.2.17 LOGS_BASE_SPEED `#define LOGS_BASE_SPEED 1`

Definition at line 30 of file entities.c.

4.23.2.18 LOGS_EXTRA_SEPARATOR `#define LOGS_EXTRA_SEPARATOR LOG_W / 2`

Definition at line 32 of file entities.c.

4.23.2.19 LOGS_MAX_USED `#define LOGS_MAX_USED 7`

Definition at line 31 of file entities.c.

4.23.2.20 LOGS_SPAWN_FRAMES `#define LOGS_SPAWN_FRAMES 60`

Definition at line 29 of file entities.c.

4.23.2.21 LOGS_SPAWN_MAX `#define LOGS_SPAWN_MAX 3`

Definition at line 28 of file entities.c.

4.23.2.22 LOGS_SPAWN_MIN `#define LOGS_SPAWN_MIN 1`

Definition at line 27 of file entities.c.

4.23.2.23 SPRITE_DEAD_TIMEOUT `#define SPRITE_DEAD_TIMEOUT 80`

Definition at line 69 of file entities.c.

4.23.2.24 SPRITE_SPLASH_RATE `#define SPRITE_SPLASH_RATE 10`

Definition at line 71 of file entities.c.

4.23.2.25 TURTLES_BASE_SPEED `#define TURTLES_BASE_SPEED 2`

Definition at line 48 of file entities.c.

4.23.2.26 TURTLES_EXTRA_SEPARATOR `#define TURTLES_EXTRA_SEPARATOR TURTLE_SIDE * 2`

Definition at line 57 of file entities.c.

4.23.2.27 TURTLES_FRAME_TIMEOUT_GOING_DOWN `#define TURTLES_FRAME_TIMEOUT_GOING_DOWN 50`

Definition at line 50 of file entities.c.

4.23.2.28 TURTLES_FRAME_TIMEOUT_GOING_UP `#define TURTLES_FRAME_TIMEOUT_GOING_UP 10`

Definition at line 52 of file entities.c.

4.23.2.29 TURTLES_FRAME_TIMEOUT_SURFACE `#define TURTLES_FRAME_TIMEOUT_SURFACE 10`

Definition at line 49 of file entities.c.

4.23.2.30 TURTLES_FRAME_TIMEOUT_WATER `#define TURTLES_FRAME_TIMEOUT_WATER 20`

Definition at line 51 of file entities.c.

4.23.2.31 TURTLES_MAX_PER_PACK `#define TURTLES_MAX_PER_PACK 2`

Definition at line 43 of file entities.c.

4.23.2.32 TURTLES_MAX_USED `#define TURTLES_MAX_USED 7`

Definition at line 47 of file entities.c.

4.23.2.33 TURTLES_MIN_PER_PACK `#define TURTLES_MIN_PER_PACK 1`

Definition at line 42 of file entities.c.

4.23.2.34 TURTLES_SPAWN_FRAMES `#define TURTLES_SPAWN_FRAMES 60`

Definition at line 44 of file entities.c.

4.23.2.35 TURTLES_SPAWN_MAX `#define TURTLES_SPAWN_MAX 2`

Definition at line 46 of file entities.c.

4.23.2.36 TURTLES_SPAWN_MIN `#define TURTLES_SPAWN_MIN 1`

Definition at line 45 of file entities.c.

4.23.2.37 TURTLES_SURFACE_FRAMES_MAX `#define TURTLES_SURFACE_FRAMES_MAX 500`

Definition at line 54 of file entities.c.

4.23.2.38 TURTLES_SURFACE_FRAMES_MIN `#define TURTLES_SURFACE_FRAMES_MIN 80`

Definition at line 53 of file entities.c.

4.23.2.39 TURTLES_WATER_FRAMES_MAX `#define TURTLES_WATER_FRAMES_MAX 100`

Definition at line 56 of file entities.c.

4.23.2.40 TURTLES_WATER_FRAMES_MIN `#define TURTLES_WATER_FRAMES_MIN 60`

Definition at line 55 of file entities.c.

4.23.3 Enumeration Type Documentation

4.23.3.1 FROG_STATES enum FROG_STATES

Definition at line 156 of file entities.c.

4.23.3.2 TURTLE_STATES enum TURTLE_STATES

Definition at line 148 of file entities.c.

4.23.4 Function Documentation

4.23.4.1 entities_draw() void entities_draw (void)

Dibuja las entidades.

Definition at line 411 of file entities.c.

4.23.4.2 entities_init() void entities_init (void)

Inicializa las entidades.

Definition at line 382 of file entities.c.

4.23.4.3 entities_move_frog() void entities_move_frog (unsigned char *direction*)

Indica que la rana debe dar un salto en la direccion dada.

Parameters

direction	enum DIRECTIONS

Definition at line 425 of file entities.c.

4.23.4.4 entities_update() void entities_update (void)

Actualiza las entidades.

Definition at line 397 of file entities.c.

4.23.5 Variable Documentation

4.23.5.1 cont `unsigned int cont`

Definition at line 183 of file entities.c.

4.23.5.2 flag `bool flag`

Definition at line 172 of file entities.c.

4.23.5.3 frame_cont `unsigned int frame_cont`

Definition at line 182 of file entities.c.

4.23.5.4 timer `unsigned int timer`

Definition at line 173 of file entities.c.

4.23.5.5 x `unsigned int x`

Definition at line 174 of file entities.c.

4.23.5.6 y `unsigned int y`

Definition at line 175 of file entities.c.

4.24 entities.c

Go to the documentation of this file.

```
00001
00012 /*****************************************************************************
00013  * INCLUDE HEADER FILES
00014  *****************************************************************************/
00015
00016 #include "entities.h"
00017 #include "allegro_stuff.h"
00018 #include "geometry.h"
00019 #include "game_data.h"
00020
00021 /*****************************************************************************
00022  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00023  *****************************************************************************/
00024
00025 //#define DEBUG_ENTITIES_TEXT
00026
00027 #define LOGS_SPAWN_MIN 1
00028 #define LOGS_SPAWN_MAX 3
00029 #define LOGS_SPAWN_FRAMES 60
00030 #define LOGS_BASE_SPEED 1
00031 #define LOGS_MAX_USED 7
00032 #define LOGS_EXTRA_SEPARATOR LOG_W / 2
00033
00034 #define CARS_SPAWN_MIN 2
00035 #define CARS_SPAWN_FRAMES 60
00036 #define CARS_BASE_SPEED 1
00037 #define CARS_MAX_USED 15
00038 #define CAR_SPEED_INCREASE 2
00039 #define CAR_WAIT_INCREASE 1
00040 #define CARS_EXTRA_SEPARATOR CAR_W * 2
00041
00042 #define TURTLES_MIN_PER_PACK 1
00043 #define TURTLES_MAX_PER_PACK 2
00044 #define TURTLES_SPAWN_FRAMES 60 // cada cuantos frames spawnean
00045 #define TURTLES_SPAWN_MIN 1     // minimas a spawnear de una
00046 #define TURTLES_SPAWN_MAX 2     // maximas a spawnear de una
00047 #define TURTLES_MAX_USED 7      // maximas en pantalla
00048 #define TURTLES_BASE_SPEED 2
00049 #define TURTLES_FRAME_TIMEOUT_SURFACE 10    // cuanto "tiempo" dura un frame dibujado antes de pasar
       al siguiente
00050 #define TURTLES_FRAME_TIMEOUT_GOING_DOWN 50 // tiempo por frame al sumergirse
00051 #define TURTLES_FRAME_TIMEOUT_WATER 20      // tiempo por frame para mostrarse bajo el agua
00052 #define TURTLES_FRAME_TIMEOUT_GOING_UP 10   // tiempo por frame para mostrarse saliendo del agua
00053 #define TURTLES_SURFACE_FRAMES_MIN 80       // minimo "tiempo" en superficie
00054 #define TURTLES_SURFACE_FRAMES_MAX 500      // maximo "tiempo" en superficie
00055 #define TURTLES_WATER_FRAMES_MIN 60         // minimo "tiempo" bajo el agua
00056 #define TURTLES_WATER_FRAMES_MAX 100        // maximo "tiempo" bajo el agua
00057 #define TURTLES_EXTRA_SEPARATOR TURTLE_SIDE * 2
00058
00059 #define COIN_SPAWN_FRAMES_MIN 300   // mínimo tiempo para respawnear coin
00060 #define COIN_SPAWN_FRAMES_MAX 600   // maximo tiempo para respawnear coin
00061 #define COIN_DESPAWN_FRAMES_MIN 700 // mínimo tiempo para sacar coin
00062 #define COIN_DESPAWN_FRAMES_MAX 900 // maximo tiempo para sacar coin
00063 #define COIN_FRAME_RATE 10          // cada cuanto gira la coin
00064 #define COIN_FRAMES_TO_WARN_A 250   // frames previos al despawneo cuando empieza a titilar
00065 #define COIN_FRAMES_TO_WARN_B 100
00066 #define COIN_WARNING_FRAMES_A 20 // blink rate
00067 #define COIN_WARNING_FRAMES_B 10
00068
00069 #define SPRITE_DEAD_TIMEOUT 80 // frames que permanece en pantalla el sprite de muerte
00070
00071 #define SPRITE_SPLASH_RATE 10 // cada cuanto avanza un frame la animacion
00072
00073 /*****************************************************************************
00074  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00075  *****************************************************************************/
00076
00077 typedef struct
00078 {
00079     int x;
00080     int y;
00081     int moving;
00082     int facing;
00083     int steps;
00084     unsigned char state;
00085     unsigned char next_action;
00086
00087 } frog_t;
00088
00089 typedef struct
00090 {
00091     int x; // Posicion del auto
00092     int y;
```

```
00093     int lane;      // Carril del auto.
00094     int dx;        // Velocidad del auto.
00095     CAR_TYPE type; // Tipo de auto.
00096     int length;    // Largo del auto.
00097     int count;
00098     bool fast;
00099     bool used; // Marca disponibilidad en el array.
00100 } car_t;
00101
00102 typedef struct
00103 {
00104     int x;
00105     int y;
00106     int lane;
00107     int dx;
00108     bool used;
00109
00110 } log_t;
00111
00112 typedef struct
00113 {
00114     int x;                          // coordenada x
00115     int y;                          // coordenada y
00116     int lane;                       // carril
00117     int dx;                         // velocidad
00118     bool used;                      // flag de usada o no
00119     unsigned char turtles_in_pack; // cantidad de tortugas en el paquete
00120
00121     struct
00122     {
00123         unsigned char frame;  // contador que indica en qué frame de la animación se está (de 1 a
     TURTLES_FRAMES)
00124         unsigned int timeout; // timeout interno para cambiar de frame
00125         unsigned int cont;    // contador interno de frames de juego ejecutados
00126     } fx;
00127
00128     int wide;            // ancho del paquete, proporcional a turtles_in_pack y a TURTLES_SIDE
00129     unsigned char state; // estado (enum TURTLE_STATES)
00130 } turtle_pack_t;
00131
00132 typedef struct
00133 {
00134     int x;
00135     int y;
00136     bool used; // flag de usada o no
00137     struct
00138     {
00139         unsigned int frame_cont;  // contador de frame a mostrar
00140         unsigned int timeout;     // Para spawneo y despawneo
00141         unsigned int blink_timer; // Para titilar coin antes de sacarla
00142         unsigned int cont;        // contador interno de frames de juego ejecutados
00143         bool flag;                // Para indicar si debe parpadear o no
00144     } fx;
00145
00146 } coin_t;
00147
00148 enum TURTLE_STATES
00149 {
00150     TURTLE_STATE_SURFACE,
00151     TURTLE_STATE_GOING_DOWN,
00152     TURTLE_STATE_WATER,
00153     TURTLE_STATE_GOING_UP
00154 };
00155
00156 enum FROG_STATES
00157 {
00158     FROG_STATE_ROAD,
00159     FROG_STATE_WATER,
00160     FROG_STATE_LOG,
00161     FROG_STATE_TURTLE,
00162     FROG_STATE_GOAL,
00163     FROG_STATE_GOAL_COIN, // meta con coin
00164     FROG_STATE_CRASH_CAR,
00165     FROG_STATE_CRASH_WALL,
00166     FROG_STATE_BOUNCING_WALL // rebota contra algun borde
00167 };
00168
00169 // Estructur para administrar el sprite de muerte
00170 static struct
00171 {
00172     bool flag;          // para indicar graficar
00173     unsigned int timer; // contador para permanecer en pantalla
00174     unsigned int x;
00175     unsigned int y;
00176 } corpse_fx;
00177
00178 // Estructura para administrar el efecto de caida en agua
```

```
00179 static struct
00180 {
00181     bool flag;               // usado o no
00182     unsigned int frame_cont; // contador de frame a mostrar
00183     unsigned int cont;       // contador de ejecucion
00184     unsigned int x;          // X topleft
00185     unsigned int y;          // Y topleft
00186
00187 } splash_fx;
00188
00189 /*********************************************************************************
00190  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00191  ********************************************************************************/
00192
00197 static void frog_init(void);
00198
00203 static void frog_update(void);
00204
00209 static void frog_draw(void);
00210
00215 static void logs_init(void);
00216
00221 static void logs_update(void);
00222
00227 static void logs_draw(void);
00228
00233 static void cars_init(void);
00234
00239 static void cars_update(void);
00240
00245 static void cars_draw(void);
00246
00251 static void turtles_init(void);
00252
00257 static void turtles_update(void);
00258
00263 static void turtles_draw(void);
00264
00269 static void coin_init(void);
00270
00275 static void coin_update(void);
00276
00281 static void coin_draw(void);
00282
00287 // static void fix_frog_pos(void);
00288
00293 static void fix_frog_coord_y(void);
00294
00303 static bool is_frog_in_goal(void);
00304
00311 static void corpse_init(int x, int y);
00312
00317 static void corpse_update(void);
00318
00323 static void corpse_draw(void);
00324
00331 static void splash_init(int x, int y);
00332
00337 static void splash_update(void);
00338
00343 static void splash_draw(void);
00344
00345 /*********************************************************************************
00346  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00347  ********************************************************************************/
00348
00349 // Rana
00350 static frog_t frog;
00351
00352 // Array de troncos
00353 static log_t log[LOGS_MAX_USED];
00354
00355 // Array de autos
00356 static car_t car[CARS_MAX_USED];
00357
00358 // Array de paquetes de tortugas
00359 static turtle_pack_t turtle_pack[TURTLES_MAX_USED];
00360
00361 // Coin
00362 static coin_t coin;
00363
00364 // Contador de frames ejecutados
00365 static unsigned long game_frames;
00366
00367 // Carriles variables.
00368 static unsigned char normal_diff_lane;
00369 static unsigned char hard_diff_lane_1;
```

```
00370 static unsigned char hard_diff_lane_2;
00371
00372 // Maximo de autos spawneados.
00373
00374 static unsigned char cars_spawn_max;
00375
00376 /*****************************************************************************
00377  *****************************************************************************
00378             GLOBAL FUNCTION DEFINITIONS
00379  *****************************************************************************
00380  *****************************************************************************/
00381
00382 void entities_init(void)
00383 {
00384     frog_init();
00385     logs_init();
00386     cars_init();
00387     turtles_init();
00388     coin_init();
00389
00390     game_frames = 0;
00391
00392     corpse_fx.flag = false;
00393
00394     splash_fx.flag = false;
00395 }
00396
00397 void entities_update()
00398 {
00399     game_frames = game_data_get_frames();
00400
00401     frog_update();
00402     logs_update();
00403     cars_update();
00404     turtles_update();
00405     coin_update();
00406
00407     corpse_update();
00408     splash_update();
00409 }
00410
00411 void entities_draw()
00412 {
00413     logs_draw();
00414     cars_draw();
00415     turtles_draw();
00416     coin_draw();
00417
00418     splash_draw();
00419     corpse_draw();
00420
00421     //"frog siempre a lo ultimo, para que se vea"
00422     frog_draw();
00423 }
00424
00425 void entities_move_frog(unsigned char direction)
00426 {
00427     if (direction == DIRECTION_DOWN || direction == DIRECTION_LEFT ||
00428         direction == DIRECTION_UP || direction == DIRECTION_RIGHT)
00429     {
00430         frog.next_action = direction;
00431     }
00432 }
00433
00434 /*****************************************************************************
00435  *****************************************************************************
00436             LOCAL FUNCTION DEFINITIONS
00437  *****************************************************************************
00438  *****************************************************************************/
00439
00440 static void frog_init(void)
00441 {
00442     frog.x = CELL_START_FROG_X;
00443     frog.y = CELL_START_FROG_Y;
00444     frog.moving = false;
00445     frog.facing = DIRECTION_UP;
00446     frog.steps = 0;
00447     frog.state = FROG_STATE_ROAD;
00448     frog.next_action = DIRECTION_NONE;
00449 }
00450
00451 static void frog_update(void)
00452 {
00453     int i;
00454
00455     bool interaction_flag = false;
00456
```

```
00457     if (!frog.moving)
00458     {
00459         if (frog.next_action == DIRECTION_DOWN || frog.next_action == DIRECTION_LEFT ||
00460             frog.next_action == DIRECTION_UP || frog.next_action == DIRECTION_RIGHT)
00461         {
00462             frog.facing = frog.next_action;
00463             frog.moving = true;
00464             frog.next_action = DIRECTION_NONE;
00465             allegro_sound_play_effect_jump();
00466         }
00467     }
00468 
00469     else if (frog.moving)
00470     {
00471 
00472         if (frog.facing == DIRECTION_LEFT)
00473             frog.x -= STEP_FRACTION_SIZE;
00474         else if (frog.facing == DIRECTION_RIGHT)
00475             frog.x += STEP_FRACTION_SIZE;
00476         else if (frog.facing == DIRECTION_UP)
00477             frog.y -= STEP_FRACTION_SIZE;
00478         else if (frog.facing == DIRECTION_DOWN)
00479             frog.y += STEP_FRACTION_SIZE;
00480 
00481         if (++frog.steps >= STEP_RATIO)
00482         {
00483             frog.steps = 0;
00484             frog.moving = false;
00485 
00486             fix_frog_coord_y();
00487         }
00488     }
00489 
00490     // donde esta parada
00491     if (!frog.moving)
00492     {
00493         unsigned int y_no_offset = frog.y - FROG_OFFSET_Y;
00494 
00495         // en alguna fila de descanso o de autos
00496         if (y_no_offset >= CELL_H * (lanes_cars[0] - 1) && y_no_offset <= DISPLAY_H - CELL_H)
00497             frog.state = FROG_STATE_ROAD;
00498 
00499         // en alguna fila de agua. Luego se actualiza si es sobre tronco o turtle
00500         else if (y_no_offset >= CELL_H * 2 && y_no_offset <= CELL_H * (lanes_cars[0] - 1))
00501             frog.state = FROG_STATE_WATER;
00502 
00503         // choque contra alguno de los muros superiores, o llegada bien a un goal
00504         else if (y_no_offset < CELL_H * 2)
00505         {
00506             if (!is_frog_in_goal())
00507             {
00508                 frog.state = FROG_STATE_CRASH_WALL;
00509             }
00510 
00511             else
00512             {
00513                 frog.state = FROG_STATE_GOAL;
00514 
00515                 // colision con coin
00516                 if (coin.used)
00517                 {
00518                     if (collideShort(coin.x,
00519                                      coin.y,
00520                                      SPRITE_COIN_SIDE,
00521                                      SPRITE_COIN_SIDE,
00522                                      frog.x,
00523                                      frog.y,
00524                                      FROG_W,
00525                                      FROG_H))
00526                     {
00527                         frog.state = FROG_STATE_GOAL_COIN;
00528                         coin.used = false;
00529                     }
00530                 }
00531             }
00532 
00533             interaction_flag = true;
00534         }
00535 
00536         /*---*/
00537 
00538         if (!interaction_flag)
00539         {
00540             // colision con autos
00541             for (i = 0; i < CARS_MAX_USED; i++)
00542             {
00543                 if (!car[i].used)
```

```
00544                     continue;
00545
00546                 if (collideShort(car[i].x,
00547                                  car[i].y,
00548                                  car[i].length,
00549                                  CAR_H,
00550                                  frog.x,
00551                                  frog.y,
00552                                  FROG_W,
00553                                  FROG_H))
00554                 {
00555                     frog.state = FROG_STATE_CRASH_CAR;
00556                     interaction_flag = true;
00557                     break; // no puede chocar con 2 autos a la vez
00558                 }
00559             }
00560         }
00561     }
00562
00563     if (!interaction_flag)
00564     {
00565         // esta en algun tronco?
00566         for (i = 0; i < LOGS_MAX_USED; i++)
00567         {
00568             if (!log[i].used)
00569                 continue;
00570
00571             if (insideShortScaled(log[i].x,
00572                                   log[i].y,
00573                                   LOG_W,
00574                                   LOG_H,
00575                                   frog.x,
00576                                   frog.y,
00577                                   FROG_W,
00578                                   FROG_H,
00579                                   INSERTION_FACTOR))
00580             {
00581                 frog.x += log[i].dx;
00582                 frog.state = FROG_STATE_LOG;
00583                 interaction_flag = true;
00584                 break; // no puede estar en 2 troncos a la vez
00585             }
00586         }
00587     }
00588
00589     if (!interaction_flag)
00590     {
00591         // esta en algun turtle_pack?
00592         for (i = 0; i < TURTLES_MAX_USED; i++)
00593         {
00594             // Omite si el pack no esta usado o si esta bajo agua
00595             if (!turtle_pack[i].used || turtle_pack[i].state == TURTLE_STATE_WATER)
00596                 continue;
00597
00598             if (insideShortScaled(turtle_pack[i].x,
00599                                   turtle_pack[i].y,
00600                                   turtle_pack[i].wide,
00601                                   TURTLE_SIDE,
00602                                   frog.x,
00603                                   frog.y,
00604                                   FROG_W,
00605                                   FROG_H,
00606                                   INSERTION_FACTOR))
00607             {
00608                 frog.x += turtle_pack[i].dx;
00609                 frog.state = FROG_STATE_TURTLE;
00610                 interaction_flag = true;
00611                 break; // no puede estar en 2 packs a la vez
00612             }
00613         }
00614     }
00615
00616     // revision de limites
00617     if (frog.x < FROG_MIN_X)
00618         frog.x = FROG_MIN_X;
00619     else if (frog.x > FROG_MAX_X)
00620         frog.x = FROG_MAX_X;
00621     else if (frog.y < FROG_MIN_Y)
00622         frog.y = FROG_MIN_Y;
00623     else if (frog.y > FROG_MAX_Y)
00624         frog.y = FROG_MAX_Y;
00625
00626     switch (frog.state)
00627     {
00628     case FROG_STATE_WATER:
00629         game_data_subtract_live();
00630         allegro_sound_play_effect_drowned();
```

```
00631
00632         splash_init(frog.x, frog.y);
00633
00634         frog_init();
00635
00636         break;
00637
00638     case FROG_STATE_CRASH_CAR:
00639         game_data_subtract_live();
00640         allegro_sound_play_effect_crash();
00641
00642         corpse_init(frog.x, frog.y);
00643
00644         frog_init();
00645
00646         break;
00647
00648     case FROG_STATE_CRASH_WALL:
00649         game_data_subtract_live();
00650         allegro_sound_play_effect_crash();
00651
00652         corpse_init(frog.x, frog.y);
00653
00654         frog_init();
00655
00656         break;
00657
00658     case FROG_STATE_GOAL:
00659         game_data_add_run_time_goal();
00660         game_data_add_score();
00661         allegro_sound_play_effect_goal();
00662
00663         frog_init();
00664
00665         break;
00666
00667     case FROG_STATE_GOAL_COIN:
00668         game_data_add_run_time_goal_bonus();
00669         game_data_add_score_bonus();
00670         allegro_sound_play_effect_bonus();
00671
00672         frog_init();
00673
00674         break;
00675
00676     default:
00677         break;
00678     }
00679
00680 #ifdef DEBUG_ENTITIES_TEXT
00681     if (!(game_frames % 10))
00682         printf("state: %d ~~ y_no_offset: %d\n", frog.state, frog.y - FROG_OFFSET_Y);
00683 #endif
00684 }
00685
00686 static void frog_draw(void)
00687 {
00688
00689     ALLEGRO_BITMAP *tempbitmap = NULL;
00690
00691     if (frog.moving)
00692     {
00693         if (frog.facing == DIRECTION_UP)
00694             tempbitmap = sprites.frog[1];
00695         if (frog.facing == DIRECTION_DOWN)
00696             tempbitmap = sprites.frog[7];
00697         if (frog.facing == DIRECTION_RIGHT)
00698             tempbitmap = sprites.frog[3];
00699         if (frog.facing == DIRECTION_LEFT)
00700             tempbitmap = sprites.frog[5];
00701     }
00702
00703     else if (!frog.moving)
00704     {
00705         if (frog.facing == DIRECTION_UP)
00706             tempbitmap = sprites.frog[0];
00707         if (frog.facing == DIRECTION_DOWN)
00708             tempbitmap = sprites.frog[6];
00709         if (frog.facing == DIRECTION_RIGHT)
00710             tempbitmap = sprites.frog[2];
00711         if (frog.facing == DIRECTION_LEFT)
00712             tempbitmap = sprites.frog[4];
00713     }
00714
00715     al_draw_bitmap(tempbitmap, frog.x, frog.y, 0);
00716
00717 #ifdef DEBUG_ENTITIES_TEXT
```

```
00718     // hitbox
00719     allegro_draw_hitbox(frog.x, frog.y, FROG_W, FROG_H);
00720     // coordenadas rana
00721     al_draw_textf(allegro_get_var_font(), al_map_rgb(200, 50, 50), 0, 0, 0, "X: %d Y: %d", frog.x,
      frog.y);
00722 #endif
00723 }
00724
00725 static void logs_init(void)
00726 {
00727     int i;
00728     for (i = 0; i < LOGS_MAX_USED; i++)
00729         log[i].used = false;
00730 }
00731
00732 static void logs_update(void)
00733 {
00734     // se busca spawnear entre LOGS_SPAWN_MIN y LOGS_SPAWN_MAX autos cada LOGS_SPAWN_FRAMES frames
00735     int new_quota = ((game_frames % LOGS_SPAWN_FRAMES) ? 0 : get_rand_between(LOGS_SPAWN_MIN,
      LOGS_SPAWN_MAX));
00736
00737     int i, used;
00738
00739     // cuento cuantos troncos usados hay
00740     for (i = 0, used = 0; i < LOGS_MAX_USED; i++)
00741         used += log[i].used;
00742
00743     for (i = 0; i < LOGS_MAX_USED; i++)
00744     {
00745         // Spawneo de troncos
00746         if (!log[i].used && new_quota > 0 && used < LOGS_MAX_USED) // Lugar libre?
00747         {
00748
00749             // Asigno carril.
00750             int temp_rand_log_lane = get_rand_between(0, LANES_LOG_TOTAL - 1);
00751             log[i].lane = lanes_logs[temp_rand_log_lane];
00752
00753             // Coordenada 'y' en funcion del carril
00754             log[i].y = CELL_H * log[i].lane + LOG_OFFSET_Y;
00755
00756             // Velocidad
00757             // log[i].dx = lanes_logs[LANES_LOG_TOTAL-1] - log[i].lane + 1;
00758             // log[i].dx = map_int(log[i].lane, 0, lanes_logs[LANES_LOG_TOTAL-1], 1, 3);
00759             log[i].dx = log[i].lane - (temp_rand_log_lane + 2) + LOGS_BASE_SPEED;
00760
00761             // en pares...
00762             if (!(log[i].lane % 2))
00763             {
00764                 // coordenada de inicio
00765                 log[i].x = -LOG_W;
00766             }
00767
00768             // en impares...
00769             else
00770             {
00771                 // coordenada de inicio
00772                 log[i].x = DISPLAY_W;
00773
00774                 // hacia el otro lado
00775                 log[i].dx *= (-1);
00776             }
00777
00778             int p;
00779             bool check; // para confirmar asignacion de lane
00780             for (p = 0, check = true; p < LOGS_MAX_USED; p++)
00781             {
00782                 // si no es el mismo tronco, y ese otro esta usado, y coinciden en lane...
00783                 if (p != i && log[p].used && log[p].lane == log[i].lane)
00784                 {
00785                     // si colisiona con algun otro tronco...
00786                     if (collide(
00787                             log[i].x - LOGS_EXTRA_SEPARATOR,
00788                             log[i].y,
00789                             log[i].x + LOG_W + LOGS_EXTRA_SEPARATOR,
00790                             log[i].y + LOG_H,
00791                             log[p].x,
00792                             log[p].y,
00793                             log[p].x + LOG_W,
00794                             log[p].y + LOG_H))
00795                     {
00796                         // no spawnea
00797                         check = false;
00798                         break;
00799                     }
00800                 }
00801             }
00802
```

```
00803             // si se puede spawnear...
00804             if (check)
00805             {
00806                 // Pasa a usado
00807                 log[i].used = true;
00808                 used++;
00809                 new_quota--;
00810             }
00811 
00812             // si no se puede spawnear...
00813             else
00814             {
00815             }
00816         }
00817 
00818         // si el tronco esta usado...
00819         else if (log[i].used)
00820         {
00821             // desplaza
00822             log[i].x += log[i].dx;
00823 
00824             // chequea si llego a los limites
00825             if ((log[i].dx > 0 && log[i].x >= DISPLAY_W) || (log[i].dx < 0 && log[i].x <= -LOG_W))
00826             {
00827                 log[i].used = false;
00828                 used--;
00829             }
00830 
00831             // printf("~log%d lane%d dx%d~\n", i, log[i].lane, log[i].dx);
00832         }
00833     }
00834 }
00835 
00836 static void logs_draw(void)
00837 {
00838     int i;
00839 
00840     for (i = 0; i < LOGS_MAX_USED; i++)
00841     {
00842         if (log[i].used)
00843         {
00844             al_draw_bitmap(sprites.log, log[i].x, log[i].y, 0);
00845 
00846 #ifdef DEBUG_ENTITIES_TEXT
00847             // hitbox
00848             allegro_draw_hitbox(log[i].x, log[i].y, LOG_W, LOG_H);
00849 #endif
00850         }
00851     }
00852 
00853 #ifdef DEBUG_ENTITIES_TEXT
00854     // coordenadas
00855     int space;
00856     for (i = 0, space = 20; i < LOGS_MAX_USED; i++, space += 10)
00857     {
00858         al_draw_textf(allegro_get_var_font(), al_map_rgb(200, 50, 50), 0, space, 0, "N°:%d X:%d Y:%d",
      i, log[i].x, log[i].y);
00859     }
00860 #endif
00861 }
00862 
00863 static void cars_init(void)
00864 {
00865     int i;
00866     // Inicio array de autos desocupando.
00867     for (i = 0; i < CARS_MAX_USED; i++)
00868         car[i].used = false;
00869 
00870     switch (game_data_get_diff())
00871     {
00872     case DIFFICULTIES_EASY:
00873         cars_spawn_max = 3;
00874         break;
00875 
00876     case DIFFICULTIES_NORMAL:
00877         normal_diff_lane = get_rand_between(lanes_cars[0], lanes_cars[LANES_CAR_TOTAL - 1]);
00878         cars_spawn_max = 4;
00879         break;
00880 
00881     case DIFFICULTIES_HARD:
00882         hard_diff_lane_1 = get_rand_between(lanes_cars[0], lanes_cars[2]);
00883         hard_diff_lane_2 = get_rand_between(lanes_cars[3], lanes_cars[4]);
00884         cars_spawn_max = 5;
00885         break;
00886     }
00887 }
00888 
```

```
00889 static void cars_update(void)
00890 {
00891     // se busca spawnear entre CARS_SPAWN_MIN y cars_spawn_max autos cada CARS_SPAWN_FRAMES frames
00892     int new_quota = ((game_frames % CARS_SPAWN_FRAMES) ? 0 : get_rand_between(CARS_SPAWN_MIN,
      cars_spawn_max));
00893
00894     int i, used;
00895
00896     // cuento cuantos autos usados hay
00897     for (i = 0, used = 0; i < CARS_MAX_USED; i++)
00898         used += car[i].used;
00899
00900     for (i = 0; i < CARS_MAX_USED; i++)
00901     {
00902         // Spawneo de autos.
00903         if (!car[i].used && new_quota > 0 && used < CARS_MAX_USED) // Lugar libre?
00904         {
00905             // Asigno carril.
00906             car[i].lane = lanes_cars[get_rand_between(0, LANES_CAR_TOTAL - 1)];
00907
00908             // Coordenada 'y' en funcion del carril
00909             car[i].y = CELL_H * car[i].lane + CAR_OFFSET_Y;
00910
00911             // Velocidad menor en rutas mas alejadas
00912             car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED;
00913             // car[i].dx = CARS_BASE_SPEED;
00914
00915             // Asigno tipos.
00916             car[i].type = get_rand_between(0, CAR_TYPE_N - 1);
00917
00918             // Defino los largos correspondientes,
00919             switch (car[i].type)
00920             {
00921             case CAR_POLICE:
00922             case CAR_YELLOW:
00923             case CAR_BLUE:
00924                 car[i].length = CAR_W;
00925                 break;
00926             case TRUCK_FIRE:
00927                 car[i].length = CAR_TRUCK_FIRE_W;
00928                 break;
00929             case TRUCK:
00930                 car[i].length = CAR_TRUCK_W;
00931                 break;
00932             default:
00933                 break;
00934             }
00935
00936             // Inicializo el contador;
00937             car[i].count = 0;
00938
00939             // Inicializo el flag.
00940             car[i].fast = 0;
00941
00942             // en pares...
00943             if (!(car[i].lane % 2))
00944             {
00945                 // coordenada de inicio
00946                 car[i].x = -car[i].length;
00947             }
00948
00949             // en impares...
00950             else
00951             {
00952                 // coordenada de inicio
00953                 car[i].x = DISPLAY_W;
00954
00955                 // hacia el otro lado
00956                 car[i].dx *= (-1);
00957             }
00958
00959             int p;
00960             bool check; // para confirmar asignacion de lane
00961             for (p = 0, check = true; p < CARS_MAX_USED; p++)
00962             {
00963                 // si no es el mismo auto, y ese otro esta usado, y coinciden en lane...
00964                 if (p != i && car[p].used && car[p].lane == car[i].lane)
00965                 {
00966                     // si colisiona con algun otro auto...
00967                     if (collide(
00968                             car[i].x - CARS_EXTRA_SEPARATOR,
00969                             car[i].y,
00970                             car[i].x + car[i].length + CARS_EXTRA_SEPARATOR, // Es el mas largo.
00971                             car[i].y + CAR_H,
00972                             car[p].x,
00973                             car[p].y,
00974                             car[p].x + car[p].length,
```

```
00975                           car[p].y + CAR_H))
00976                   {
00977                       // no spawnea
00978                       check = false;
00979                       break;
00980                   }
00981               }
00982           }
00983
00984           // si se puede spawnear...
00985           if (check)
00986           {
00987               // Pasa a usado
00988               car[i].used = true;
00989               used++;
00990               new_quota--;
00991           }
00992
00993           // si no se puede spawnear...
00994           else
00995           {
00996           }
00997       }
00998
00999       // si el auto esta usado...
01000       else if (car[i].used)
01001       {
01002           // Carril con velocidad variable
01003           if (car[i].count < CAR_WAIT_INCREASE)
01004           {
01005               switch (game_data_get_diff())
01006               {
01007               case DIFFICULTIES_EASY:
01008                   break;
01009
01010               case DIFFICULTIES_NORMAL:
01011                   if (car[i].lane == normal_diff_lane)
01012                   {
01013                       if (!(game_frames % FPS))
01014                       {
01015                           if (car[i].fast == 0)
01016                           {
01017                               car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) +
    CARS_BASE_SPEED + CAR_SPEED_INCREASE;
01018                               if (car[i].lane % 2)
01019                                   car[i].dx *= (-1);
01020                               car[i].fast = 1;
01021                           }
01022                           else
01023                           {
01024                               car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) +
    CARS_BASE_SPEED;
01025                               if (car[i].lane % 2)
01026                                   car[i].dx *= (-1);
01027                               car[i].fast = 0;
01028                           }
01029                       }
01030                   }
01031                   break;
01032               case DIFFICULTIES_HARD:
01033                   if ((car[i].lane == hard_diff_lane_1) || (car[i].lane == hard_diff_lane_2))
01034                   {
01035                       if (!(game_frames % FPS))
01036                       {
01037                           if (car[i].fast == 0)
01038                           {
01039                               car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) +
    CARS_BASE_SPEED + CAR_SPEED_INCREASE;
01040                               if (car[i].lane % 2)
01041                                   car[i].dx *= (-1);
01042                               car[i].fast = 1;
01043                           }
01044                           else
01045                           {
01046                               car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) +
    CARS_BASE_SPEED;
01047                               if (car[i].lane % 2)
01048                                   car[i].dx *= (-1);
01049                               car[i].fast = 0;
01050                           }
01051                       }
01052                   }
01053               default:
01054                   break;
01055               }
01056           }
01057           else
```

```
01058                 car[i].count++;
01059
01060             // Desplazamiento
01061             car[i].x += car[i].dx;
01062
01063             // chequea si llego a los limites
01064             if ((car[i].dx > 0 && car[i].x >= DISPLAY_W) || (car[i].dx < 0 && car[i].x <=
      -car[i].length))
01065             {
01066                 car[i].used = false;
01067                 used--;
01068             }
01069
01070             // printf("~car%d lane%d dx%d~\n", i, car[i].lane, car[i].dx);
01071         }
01072     }
01073 }
01074
01075 static void cars_draw()
01076 {
01077     int i;
01078     bool flag;
01079
01080     ALLEGRO_BITMAP *temp_bitmap = NULL;
01081
01082     for (i = 0; i < CARS_MAX_USED; i++)
01083     {
01084         if (car[i].used)
01085         {
01086             if (car[i].dx < 0)
01087                 flag = ALLEGRO_FLIP_HORIZONTAL;
01088             else
01089                 flag = 0;
01090
01091             temp_bitmap = sprites.car[car[i].type];
01092
01093             // Dibujo los autos en sus carriles.
01094             al_draw_bitmap(temp_bitmap, car[i].x, car[i].y, flag);
01095
01096 #ifdef DEBUG_ENTITIES_TEXT
01097             // Dibujo hitbox
01098             allegro_draw_hitbox(car[i].x, car[i].y, car[i].length, CAR_H);
01099 #endif
01100         }
01101     }
01102
01103 #ifdef DEBUG_ENTITIES_TEXT
01104     // coordenadas
01105     int space;
01106     for (i = 0, space = 200; i < CARS_MAX_USED; i++, space += 20)
01107     {
01108         // al_draw_textf(allegro_get_var_font(), al_map_rgb(255, 255, 255), 0, space, 0, "N°:%d X:%d
      Y:%d dx:%d", i, car[i].x, car[i].y, car[i].dx);
01109         al_draw_textf(allegro_get_var_font(), al_map_rgb(255, 255, 255), 0, space, 0, "Lane:%d dx:%d",
      car[i].lane, car[i].dx);
01110     }
01111 #endif
01112 }
01113
01114 static void turtles_init(void)
01115 {
01116     int i;
01117     for (i = 0; i < TURTLES_MAX_USED; i++)
01118     {
01119         turtle_pack[i].used = false;
01120     }
01121 }
01122
01123 static void turtles_update(void)
01124 {
01125     int new_quota = ((game_frames % TURTLES_SPAWN_FRAMES) ? 0 : get_rand_between(TURTLES_SPAWN_MIN,
      TURTLES_SPAWN_MAX));
01126
01127     int i, used;
01128
01129     for (i = 0, used = 0; i < TURTLES_MAX_USED; i++)
01130     {
01131         if (turtle_pack[i].used)
01132             used++;
01133     }
01134
01135     for (i = 0; i < TURTLES_MAX_USED; i++)
01136     {
01137         // Spawneo de turtle_packs
01138         if (!turtle_pack[i].used && new_quota > 0 && used < TURTLES_MAX_USED) // Lugar libre?
01139         {
01140
```

```
01141             // defino tortugas en el pack
01142             turtle_pack[i].turtles_in_pack = get_rand_between(TURTLES_MIN_PER_PACK,
      TURTLES_MAX_PER_PACK);
01143
01144             // calculo ancho del pack
01145             turtle_pack[i].wide = TURTLE_SIDE * turtle_pack[i].turtles_in_pack;
01146
01147             // Asigno carril.
01148             turtle_pack[i].lane = lanes_turtles[get_rand_between(0, LANES_TURTLE_TOTAL - 1)];
01149
01150             // Coordenada 'y' en funcion del carril
01151             turtle_pack[i].y = CELL_H * turtle_pack[i].lane;
01152
01153             // Velocidad
01154             // turtle_pack[i].dx = lanes_turtles[LANES_TURTLE_TOTAL- turtle_pack[i].lane + 1];
01155             turtle_pack[i].dx = TURTLES_BASE_SPEED;
01156
01157             // en pares...
01158             if (!(turtle_pack[i].lane % 2))
01159             {
01160                 // coordenada de inicio
01161                 turtle_pack[i].x = -turtle_pack[i].wide;
01162             }
01163
01164             // en impares...
01165             else
01166             {
01167                 // coordenada de inicio
01168                 turtle_pack[i].x = DISPLAY_W;
01169
01170                 // hacia el otro lado
01171                 turtle_pack[i].dx *= (-1);
01172             }
01173
01174             int p;
01175             bool check; // para confirmar asignacion de lane
01176             for (p = 0, check = true; p < TURTLES_MAX_USED; p++)
01177             {
01178                 // si no es el mismo pack, y ese otro esta usado, y coinciden en lane...
01179                 if (p != i && turtle_pack[p].used && turtle_pack[p].lane == turtle_pack[i].lane)
01180                 {
01181                     // si colisiona con algun otro pack...
01182                     if (collide(
01183                             turtle_pack[i].x - TURTLES_EXTRA_SEPARATOR,
01184                             turtle_pack[i].y,
01185                             turtle_pack[i].x + turtle_pack[i].wide + TURTLES_EXTRA_SEPARATOR,
01186                             turtle_pack[i].y + TURTLE_SIDE,
01187                             turtle_pack[p].x,
01188                             turtle_pack[p].y,
01189                             turtle_pack[p].x + turtle_pack[p].wide,
01190                             turtle_pack[p].y + TURTLE_SIDE))
01191                     {
01192                         // no spawnea
01193                         check = false;
01194                         break;
01195                     }
01196                 }
01197             }
01198
01199             // si se puede spawnear...
01200             if (check)
01201             {
01202                 // Pasa a usado
01203                 turtle_pack[i].used = true;
01204                 used++;
01205
01206                 // se inicializa el contador de frames
01207                 turtle_pack[i].fx.frame = 0;
01208                 turtle_pack[i].fx.cont = 1;
01209                 turtle_pack[i].fx.timeout = 0;
01210                 // fuera del agua
01211                 turtle_pack[i].state = TURTLE_STATE_SURFACE;
01212
01213                 new_quota--;
01214             }
01215
01216             // si no se puede spawnear...
01217             else
01218             {
01219             }
01220         }
01221
01222         // si el pack esta usado...
01223         else if (turtle_pack[i].used)
01224         {
01225             // desplaza
01226             turtle_pack[i].x += turtle_pack[i].dx;
```

```
01227
01228           switch (turtle_pack[i].state)
01229           {
01230           case TURTLE_STATE_SURFACE:
01231               if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_SURFACE))
01232                   turtle_pack[i].fx.frame++;
01233
01234               // si no esta inicializado, inicializo timeout
01235               if (!turtle_pack[i].fx.timeout)
01236                   turtle_pack[i].fx.timeout = get_rand_between(TURTLES_SURFACE_FRAMES_MIN,
    TURTLES_SURFACE_FRAMES_MAX);
01237
01238               // pasa a agua
01239               if (!(turtle_pack[i].fx.cont % turtle_pack[i].fx.timeout))
01240               {
01241                   turtle_pack[i].state = TURTLE_STATE_GOING_DOWN;
01242                   turtle_pack[i].fx.frame = 7;
01243                   turtle_pack[i].fx.timeout = 0;
01244                   turtle_pack[i].fx.cont = 1;
01245               }
01246
01247               // Reinicia animacion
01248               else if (turtle_pack[i].fx.frame == 7)
01249                   turtle_pack[i].fx.frame = 0;
01250
01251               break;
01252
01253           case TURTLE_STATE_GOING_DOWN:
01254               if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_GOING_DOWN))
01255                   turtle_pack[i].fx.frame++;
01256
01257               if (turtle_pack[i].fx.frame == 9)
01258               {
01259                   turtle_pack[i].state = TURTLE_STATE_WATER;
01260                   turtle_pack[i].fx.cont = 1;
01261               }
01262
01263               break;
01264
01265           case TURTLE_STATE_WATER:
01266               if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_WATER))
01267                   turtle_pack[i].fx.frame++;
01268
01269               // si no esta inicializado, inicializo timeout
01270               if (!turtle_pack[i].fx.timeout)
01271                   turtle_pack[i].fx.timeout = get_rand_between(TURTLES_WATER_FRAMES_MIN,
    TURTLES_WATER_FRAMES_MAX);
01272
01273               // pasa a fuera
01274               if (!(turtle_pack[i].fx.cont % turtle_pack[i].fx.timeout))
01275               {
01276                   turtle_pack[i].state = TURTLE_STATE_GOING_UP;
01277                   turtle_pack[i].fx.frame = 10;
01278                   turtle_pack[i].fx.timeout = 0;
01279                   turtle_pack[i].fx.cont = 1;
01280               }
01281
01282               // Reinicia animacion
01283               else if (turtle_pack[i].fx.frame == 11)
01284                   turtle_pack[i].fx.frame = 9;
01285
01286               break;
01287
01288           case TURTLE_STATE_GOING_UP:
01289               if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_GOING_UP))
01290                   turtle_pack[i].fx.frame--;
01291
01292               if (turtle_pack[i].fx.frame == 7)
01293               {
01294                   turtle_pack[i].fx.frame = 6;
01295                   turtle_pack[i].state = TURTLE_STATE_SURFACE;
01296                   turtle_pack[i].fx.cont = 1;
01297               }
01298
01299               break;
01300
01301           default:
01302               break;
01303           }
01304
01305           // chequea si llego a los limites
01306           if ((turtle_pack[i].dx > 0 && turtle_pack[i].x >= DISPLAY_W) || (turtle_pack[i].dx < 0 &&
    turtle_pack[i].x <= -turtle_pack[i].wide))
01307           {
01308               turtle_pack[i].used = false;
01309               used--;
01310           }
```

```
01311
01312             // printf("~turtle_pack%d lane%d dx%d~\n", i, turtle_pack[i].lane, turtle_pack[i].dx);
01313         }
01314     }
01315 }
01316
01317 static void turtles_draw(void)
01318 {
01319     int i, j, flag;
01320     for (i = 0; i < TURTLES_MAX_USED; i++)
01321     {
01322         if (turtle_pack[i].used)
01323         {
01324             for (j = 0; j < turtle_pack[i].turtles_in_pack; j++)
01325             {
01326                 if (turtle_pack[i].dx < 0)
01327                     flag = ALLEGRO_FLIP_HORIZONTAL;
01328                 else
01329                     flag = 0;
01330
01331                 al_draw_bitmap(sprites.turtle[turtle_pack[i].fx.frame], turtle_pack[i].x + TURTLE_SIDE
      * j, turtle_pack[i].y, flag);
01332             }
01333
01334 #ifdef DEBUG_ENTITIES_TEXT
01335             // Dibujo hitbox
01336             allegro_draw_hitbox(turtle_pack[i].x, turtle_pack[i].y, turtle_pack[i].wide, TURTLE_SIDE);
01337 #endif
01338         }
01339     }
01340
01341 #ifdef DEBUG_ENTITIES_TEXT
01342     // coordenadas
01343     int space;
01344     for (i = 0, space = 350; i < TURTLES_MAX_USED; i++, space += 10)
01345     {
01346         al_draw_textf(allegro_get_var_font(), al_map_rgb(200, 50, 50), 0, space, 0, "N°:%d X:%d Y:%d",
      i, turtle_pack[i].x, turtle_pack[i].y);
01347     }
01348 #endif
01349 }
01350
01351 static void coin_init(void)
01352 {
01353     coin.used = false;
01354     coin.y = CELL_H + SPRITE_COIN_OFFSET_XY + GOAL_ROW_OFFSET_Y_FIX;
01355
01356     coin.fx.blink_timer = 0;
01357     coin.fx.timeout = 0;
01358     coin.fx.flag = false;
01359     coin.fx.cont = 1;
01360 }
01361
01362 static void coin_update(void)
01363 {
01364     if (!coin.used)
01365     {
01366         // si no esta inicializado, inicializo timeout para spawneo
01367         if (!coin.fx.timeout)
01368             coin.fx.timeout = get_rand_between(COIN_SPAWN_FRAMES_MIN, COIN_SPAWN_FRAMES_MAX);
01369
01370         if (!(coin.fx.cont % coin.fx.timeout))
01371         {
01372             // calculo de coordenada x para alguno de los puntos de llegada
01373             int temp_goal = get_rand_between(0, MAX_GOALS - 1);
01374
01375             // si el goal está libre...
01376             if (!game_data_get_goal_state(temp_goal))
01377             {
01378                 allegro_sound_play_effect_coin_drop();
01379
01380                 coin.x = CELL_W * goal_cols[temp_goal] + SPRITE_COIN_OFFSET_XY - 1;
01381                 // marcado como usado
01382                 coin.used = true;
01383                 // desinicializo el timeout
01384                 coin.fx.timeout = 0;
01385
01386                 coin.fx.blink_timer = 0;
01387                 coin.fx.cont = 1;
01388                 coin.fx.frame_cont = 0;
01389             }
01390
01391             // si no, cuando pasa otro timeout se intenta de nuevo
01392             else
01393             {
01394             }
01395         }
```

```
01396     }
01397
01398     else
01399     {
01400         // timeout para despawneo
01401         if (!coin.fx.timeout)
01402             coin.fx.timeout = get_rand_between(COIN_DESPAWN_FRAMES_MIN, COIN_DESPAWN_FRAMES_MAX);
01403
01404         if (++coin.fx.blink_timer > coin.fx.timeout - COIN_FRAMES_TO_WARN_A)
01405         {
01406             if (coin.fx.blink_timer > coin.fx.timeout - COIN_FRAMES_TO_WARN_B)
01407             {
01408                 if (!(coin.fx.cont % COIN_WARNING_FRAMES_B))
01409                     coin.fx.flag = !coin.fx.flag;
01410             }
01411             else
01412             {
01413                 if (!(coin.fx.cont % COIN_WARNING_FRAMES_A))
01414                     coin.fx.flag = !coin.fx.flag;
01415             }
01416         }
01417
01418         if (!(game_frames % COIN_FRAME_RATE))
01419         {
01420             if (++coin.fx.frame_cont == SPRITE_COIN_FRAMES)
01421                 coin.fx.frame_cont = 0;
01422         }
01423
01424         // si se puede despawnear
01425         if (!(coin.fx.cont % coin.fx.timeout))
01426         {
01427             // coin no usada
01428             coin.used = false;
01429
01430             // desinicializo timeout
01431             coin.fx.timeout = 0;
01432
01433             // saco el blinking
01434             coin.fx.flag = false;
01435
01436             coin.fx.cont = 1;
01437         }
01438     }
01439
01440     coin.fx.cont++;
01441 }
01442
01443 static void coin_draw(void)
01444 {
01445     if (coin.used)
01446     {
01447         // Si no está el flag, dibujo sprite normalmente
01448         if (!coin.fx.flag)
01449             al_draw_bitmap(sprites.coin.frame[coin.fx.frame_cont], coin.x, coin.y, 0);
01450     }
01451
01452 #ifdef DEBUG_ENTITIES_TEXT
01453
01454     // hitbox
01455     allegro_draw_hitbox(coin.x, coin.y, COIN_SIDE, COIN_SIDE);
01456
01457     // coordenadas
01458     int space = 500;
01459     al_draw_textf(allegro_get_var_font(), al_map_rgb(200, 50, 50), 0, space, 0, "Coin ~ X:%d Y:%d",
       coin.x, coin.y);
01460 #endif
01461 }
01462
01463 static void fix_frog_coord_y(void)
01464 {
01465     int y = (frog.y - FROG_OFFSET_Y);
01466
01467     int y_values[ROWS];
01468
01469     int i;
01470
01471     // Carga valores "correctos" de y
01472     for (i = 1; i < ROWS - 1; i++)
01473         y_values[i] = i * CELL_H;
01474
01475     int temp_a, temp_b;
01476     for (i = 1; i < ROWS - 1; i++)
01477     {
01478         temp_a = y - y_values[i];
01479
01480         if (temp_a > 0)
01481             continue;
```

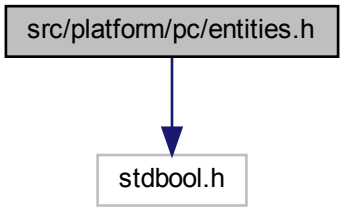
```
01482         if (temp_a == 0)
01483             break;
01484
01485         temp_b = y_values[i - 1] - y;
01486
01487         //"si está más cerca de la fila 'i' que de la 'i+1"
01488         if (temp_a <= temp_b)
01489             frog.y = y_values[i - 1] + FROG_OFFSET_Y;
01490         else
01491             frog.y = y_values[i] + FROG_OFFSET_Y;
01492
01493         break;
01494     }
01495 }
01496
01497 static bool is_frog_in_goal(void)
01498 {
01499     bool state = false;
01500     int x = frog.x;
01501
01502     int i, x_col;
01503     for (i = 0; i < MAX_GOALS; i++)
01504     {
01505         // Coordenada top left del punto de llegada
01506         x_col = goal_cols[i] * CELL_W;
01507
01508         // Calculo para ver si entro bien o no
01509         if ((x > x_col - GOAL_ROW_MARGIN_TO_REACH) &&
01510             ((x + FROG_W) < x_col + CELL_W + GOAL_ROW_MARGIN_TO_REACH))
01511         {
01512             // coodenada X aceptable
01513             state = true;
01514             break;
01515         }
01516     }
01517
01518     // Si coincide en coordenada y el goal esta libre...
01519     if (state && !game_data_get_goal_state(i))
01520     {
01521         // marca el goal como completo
01522         game_data_set_goal(i);
01523     }
01524     else
01525     {
01526         // no llego a un goal valido
01527         state = false;
01528     }
01529
01530     return state;
01531 }
01532
01533 static void corpse_init(int x, int y)
01534 {
01535     corpse_fx.flag = true;
01536     corpse_fx.timer = 1;
01537     corpse_fx.x = x - FROG_OFFSET_X + SPRITE_DEAD_OFFSET;
01538     corpse_fx.y = y - FROG_OFFSET_Y + SPRITE_DEAD_OFFSET;
01539 }
01540
01541 static void corpse_update(void)
01542 {
01543     if (corpse_fx.flag)
01544     {
01545         if (!(corpse_fx.timer++ % SPRITE_DEAD_TIMEOUT))
01546             corpse_fx.flag = false;
01547     }
01548 }
01549
01550 static void corpse_draw(void)
01551 {
01552     if (corpse_fx.flag)
01553         al_draw_bitmap(sprites.dead, corpse_fx.x, corpse_fx.y, 0);
01554 }
01555
01556 static void splash_init(int x, int y)
01557 {
01558     splash_fx.flag = true;
01559     splash_fx.cont = 1;
01560     splash_fx.frame_cont = 0;
01561     splash_fx.x = x - FROG_OFFSET_X + SPRITE_SPLASH_OFFSET_X;
01562     splash_fx.y = y - FROG_OFFSET_Y + SPRITE_SPLASH_OFFSET_Y;
01563 }
01564
01565 static void splash_update(void)
01566 {
01567     if (splash_fx.flag)
01568     {
```

```
01569         if (!(splash_fx.cont % SPRITE_SPLASH_RATE))
01570         {
01571             if (++splash_fx.frame_cont == SPRITE_SPLASH_FRAMES)
01572             {
01573                 splash_fx.frame_cont = 0;
01574                 splash_fx.flag = false;
01575             }
01576         }
01577 
01578         splash_fx.cont++;
01579     }
01580 }
01581 
01582 static void splash_draw(void)
01583 {
01584     if (splash_fx.flag)
01585         al_draw_bitmap(sprites.splash.frame[splash_fx.frame_cont], splash_fx.x, splash_fx.y, 0);
01586 }
```

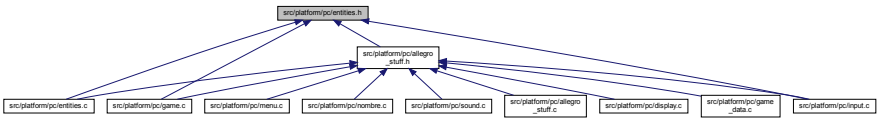
4.25 src/platform/pc/entities.h File Reference

Header del modulo entities. Prototipos de funciones globales para el manejo de entidades.

`#include <stdbool.h>`
Include dependency graph for entities.h:



This graph shows which files directly or indirectly include this file:



Functions

- void entities_init (void)

  Inicializa las entidades.
- void entities_update (void)

  Actualiza las entidades.
- void entities_draw (void)

  Dibuja las entidades.
- void entities_move_frog (unsigned char direction)

  Indica que la rana debe dar un salto en la direccion dada.

4.25.1 Detailed Description

Header del modulo entities. Prototipos de funciones globales para el manejo de entidades.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file entities.h.

4.25.2 Function Documentation

4.25.2.1 entities_draw() `void entities_draw ( void )`

Dibuja las entidades.

Definition at line 411 of file entities.c.

4.25.2.2 entities_init() `void entities_init ( void )`

Inicializa las entidades.

Definition at line 382 of file entities.c.

4.25.2.3 entities_move_frog() `void entities_move_frog ( unsigned char direction )`

Indica que la rana debe dar un salto en la direccion dada.

Parameters

direction	enum DIRECTIONS

Definition at line 425 of file entities.c.

4.25.2.4 entities_update() void entities_update (void)

Actualiza las entidades.

Definition at line 397 of file entities.c.

4.26 entities.h

Go to the documentation of this file.

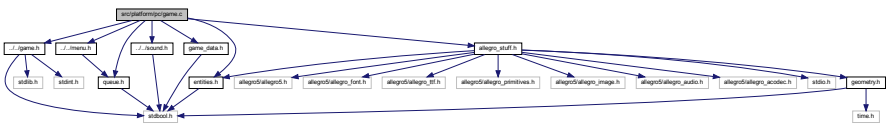
```
00001
00012 #ifndef _ENTITIES_H_
00013 #define _ENTITIES_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include <stdbool.h>
00020
00021 /*****************************************************************************
00022  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00023  *****************************************************************************/
00024
00029 void entities_init(void);
00030
00035 void entities_update(void);
00036
00041 void entities_draw(void);
00042
00048 void entities_move_frog(unsigned char direction);
00049
00050 /*****************************************************************************
00051  *****************************************************************************/
00052
00053 #endif // _ENTITIES_H_
```

4.27 src/platform/pc/game.c File Reference

Source del modulo game, orientado a PC. Vincula la FSM con lo específico de PC en lo relacionado a la interacción con el juego.

```
#include "../../game.h"
#include "../../menu.h"
#include "../../queue.h"
#include "../../sound.h"
#include "game_data.h"
#include "entities.h"
#include "allegro_stuff.h"
```

Include dependency graph for game.c:



Functions

- void setNombre (char *nombre)
  Confirma el nombre del jugador.
- void setMaxPuntos (unsigned long long max)
  Setea los puntos maximos del jugador.
- void setDificultad (int diff)
  Setea la dificultad a usar.
- bool tiempoRefrescoEntidades (void)
  Chequea si es tiempo de refrescar entidades según la plataforma.
- char * getNombre (void)
  Devuelve el nombre del jugador.
- unsigned long long getPuntos (void)
  Devuelve el puntaje del jugador.
- unsigned long long getMaxPuntos (void)
  Devuelve el puntaje máximo del jugador.
- int getNivel (void)
  Devuelve el nivel//run del jugador.
- void inicializarJuego (void)
  Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.
- void pausarJuego (void)
  Pausa el juego.
- void reiniciarNivel (void)
  Configuraciones para reiniciar el nivel.
- void refrescar (void)
  Actualizaciones relativas a actualizar las entidades.
- void moverAdelante (void)
  Avanza el jugador.
- void moverAtras (void)
  Retrocede el jugador.
- void moverIzda (void)
  Mueve el jugador a la izquierda.
- void moverDcha (void)
  Mueve el jugador a la derecha.
- void respawn (void)
  Respawnea el jugador.
- void actualizarInterfaz (void)
  Actualizaciones relativas a lo visual.
- void reanudarJuego (void)
  Saca el juego de pausa.

4.27.1 Detailed Description

Source del modulo game, orientado a PC. Vincula la FSM con lo específico de PC en lo relacionado a la interacción con el juego.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file game.c.

4.27.2 Function Documentation

4.27.2.1 actualizarInterfaz() `void actualizarInterfaz ( void )`

Actualizaciones relativas a lo visual.

Definition at line 155 of file game.c.

4.27.2.2 getMaxPuntos() `unsigned long long getMaxPuntos ( void )`

Devuelve el puntaje máximo del jugador.

Returns

unsigned long long

Definition at line 82 of file game.c.

4.27.2.3 getNivel() `int getNivel ( void )`

Devuelve el nivel//run del jugador.

Returns

int

Definition at line 87 of file game.c.

4.27.2.4 getNombre() `char * getNombre ( void )`

Devuelve el nombre del jugador.

Returns

char*

Definition at line 72 of file game.c.

4.27.2.5 getPuntos() unsigned long long getPuntos (void)

Devuelve el puntaje del jugador.

Returns

unsigned long long

Definition at line 77 of file game.c.

4.27.2.6 inicializarJuego() void inicializarJuego (void)

Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.

Definition at line 92 of file game.c.

4.27.2.7 moverAdelante() void moverAdelante (void)

Avanza el jugador.

Definition at line 131 of file game.c.

4.27.2.8 moverAtras() void moverAtras (void)

Retrocede el jugador.

Definition at line 136 of file game.c.

4.27.2.9 moverDcha() void moverDcha (void)

Mueve el jugador a la derecha.

Definition at line 146 of file game.c.

4.27.2.10 moverIzda() `void moverIzda (`

`void )`

Mueve el jugador a la izquierda.

Definition at line 141 of file game.c.

4.27.2.11 pausarJuego() `void pausarJuego (`

`void )`

Pausa el juego.

Definition at line 101 of file game.c.

4.27.2.12 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 188 of file game.c.

4.27.2.13 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 114 of file game.c.

4.27.2.14 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 105 of file game.c.

4.27.2.15 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 151 of file game.c.

4.27.2.16 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 47 of file game.c.

4.27.2.17 setMaxPuntos() `void setMaxPuntos ( unsigned long long max )`

Setea los puntos maximos del jugador.

Parameters

max	

Definition at line 42 of file game.c.

4.27.2.18 setNombre() `void setNombre ( char * nombre )`

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 37 of file game.c.

4.27.2.19 tiempoRefrescoEntidades() `bool tiempoRefrescoEntidades ( void )`

Chequea si es tiempo de refrescar entidades según la plataforma.

Returns

true

false

Definition at line 67 of file game.c.

4.28 game.c

Go to the documentation of this file.

```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "../../game.h"
00017 #include "../../menu.h"
00018 #include "../../queue.h"
00019 #include "../../sound.h"
00020
00021 #include "game_data.h"
00022 #include "entities.h"
00023 #include "allegro_stuff.h"
00024
00025 /*******************************************************************************
00026  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00027  ******************************************************************************/
00028
00029 static bool next_run_flag = false;
00030
00031 /*******************************************************************************
00032  *******************************************************************************
00033                         GLOBAL FUNCTION DEFINITIONS
00034  *******************************************************************************
00035  ******************************************************************************/
00036
00037 void setNombre(char *nombre)
00038 {
00039     game_data_overwrite_name(nombre);
00040 }
00041
00042 void setMaxPuntos(unsigned long long max)
00043 {
00044     game_data_set_score_max(max);
00045 }
00046
00047 void setDificultad(int diff)
00048 {
00049     switch (diff)
00050     {
00051     case 0:
00052         game_data_set_diff(DIFFICULTIES_EASY);
00053         break;
00054
00055     case 1:
00056         game_data_set_diff(DIFFICULTIES_NORMAL);
00057         break;
00058
00059     case 2:
00060         game_data_set_diff(DIFFICULTIES_HARD);
00061
00062     default:
00063         break;
00064     }
00065 }
00066
00067 bool tiempoRefrescoEntidades(void)
00068 {
00069     return allegro_get_var_redraw();
00070 }
00071
00072 char *getNombre(void)
00073 {
00074     return game_data_get_name();
00075 }
00076
00077 unsigned long long getPuntos(void)
00078 {
00079     return game_data_get_score();
00080 }
00081
00082 unsigned long long getMaxPuntos(void)
00083 {
00084     return game_data_get_score_max();
00085 }
00086
00087 int getNivel(void)
00088 {
00089     return game_data_get_run_number();
00090 }
00091
00092 void inicializarJuego(void)
00093 {
```

```
00094     game_data_init();
00095     entities_init();
00096
00097     allegro_clear_display();
00098     al_flip_display();
00099 }
00100
00101 void pausarJuego(void)
00102 {
00103 }
00104
00105 void reiniciarNivel(void)
00106 {
00107     if (next_run_flag)
00108     {
00109         game_data_next_run();
00110         next_run_flag = false;
00111     }
00112 }
00113
00114 void refrescar(void)
00115 {
00116     game_data_update();
00117     entities_update();
00118
00119     if (game_data_are_goals_full())
00120     {
00121         next_run_flag = true;
00122
00123         reproducirEfecto(EFECTO_NIVEL_COMPLETO);
00124         reiniciarNivel();
00125     }
00126
00127     if (game_data_get_game_over_flag())
00128         queueInsertar(GAME_OVER);
00129 }
00130
00131 void moverAdelante(void)
00132 {
00133     entities_move_frog(DIRECTION_UP);
00134 }
00135
00136 void moverAtras(void)
00137 {
00138     entities_move_frog(DIRECTION_DOWN);
00139 }
00140
00141 void moverIzda(void)
00142 {
00143     entities_move_frog(DIRECTION_LEFT);
00144 }
00145
00146 void moverDcha(void)
00147 {
00148     entities_move_frog(DIRECTION_RIGHT);
00149 }
00150
00151 void respawn(void)
00152 {
00153 }
00154
00155 void actualizarInterfaz(void)
00156 {
00157     if (allegro_get_last_key() == ALLEGRO_KEY_8 && !allegro_get_rick_flag())
00158     {
00159         allegro_rick_on();
00160         allegro_set_rick_flag(true);
00161     }
00162
00163     if (allegro_get_last_key() == ALLEGRO_KEY_9 && allegro_get_rick_flag())
00164     {
00165         allegro_rick_off();
00166         allegro_set_rick_flag(false);
00167     }
00168
00169     if (allegro_get_var_redraw())
00170     {
00171         allegro_clear_display();
00172         allegro_draw_background();
00173
00174         if (allegro_get_rick_flag())
00175             allegro_rick_draw();
00176
00177         entities_draw();
00178         game_data_draw();
00179
00180         al_draw_bitmap(sprites.border, SPRITE_BORDER_START_X, SPRITE_BORDER_START_Y, 0);
```

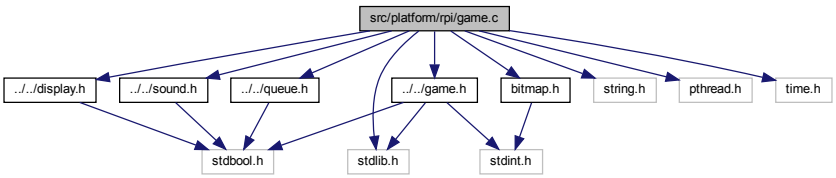
```
00181
00182         al_flip_display();
00183
00184         allegro_set_var_redraw(false);
00185     }
00186 }
00187
00188 void reanudarJuego(void)
00189 {
00190 }
```

4.29 src/platform/rpi/game.c File Reference

Archivo para manejar la información del juego.

```
#include "../../game.h"
#include "bitmap.h"
#include "../../display.h"
#include "../../sound.h"
#include "../../queue.h"
#include <string.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>
```

Include dependency graph for game.c:



Macros

- #define POS_AUTOS_INICIO 4
- #define POS_AUTOS_FIN 13
- #define CANT_CARRILES 5
- #define L_MAX 64

Functions

- void setNombre (char ∗nombre)

  Confirma el nombre del jugador.
- void setMaxPuntos (unsigned long long max)

  Setea los puntos maximos del jugador.
- void limpiarMapa ()
- void moverCarrriles (int x)
- void spawnearAutos ()
- void actualizarMapa ()
- void refrescar ()

  Actualizaciones relativas a actualizar las entidades.

- bool tiempoRefrescoEntidades (void)

  Chequea si es tiempo de refrescar entidades según la plataforma.
- void setDificultad (int dificultad)

  Setea la dificultad a usar.
- char * getNombre ()

  Devuelve el nombre del jugador.
- unsigned long long getPuntos ()

  Devuelve el puntaje del jugador.
- unsigned long long getMaxPuntos ()

  Devuelve el puntaje máximo del jugador.
- int getNivel ()

  Devuelve el nivel//run del jugador.
- void reiniciarNivel ()

  Configuraciones para reiniciar el nivel.
- void respawn ()

  Respawnea el jugador.
- void moverAdelante ()

  Avanza el jugador.
- void moverAtras ()

  Retrocede el jugador.
- void moverIzda ()

  Mueve el jugador a la izquierda.
- void moverDcha ()

  Mueve el jugador a la derecha.
- void perderVida ()

  Resta una vida.
- void inicializarJuego ()

  Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.
- void pausarJuego ()

  Pausa el juego.
- void actualizarInterfaz ()

  Actualizaciones relativas a lo visual.
- void reanudarJuego (void)

  Saca el juego de pausa.

Variables

- matriz_t disp_matriz

4.29.1 Detailed Description

Archivo para manejar la información del juego.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file game.c.

4.29.2 Macro Definition Documentation

4.29.2.1 CANT_CARRILES `#define CANT_CARRILES 5`

Definition at line 30 of file game.c.

4.29.2.2 L_MAX `#define L_MAX 64`

Definition at line 31 of file game.c.

4.29.2.3 POS_AUTOS_FIN `#define POS_AUTOS_FIN 13`

Definition at line 29 of file game.c.

4.29.2.4 POS_AUTOS_INICIO `#define POS_AUTOS_INICIO 4`

Definition at line 28 of file game.c.

4.29.3 Function Documentation

4.29.3.1 actualizarInterfaz() `void actualizarInterfaz ( void )`

Actualizaciones relativas a lo visual.

Definition at line 341 of file game.c.

4.29.3.2 actualizarMapa() `void actualizarMapa ( )`

Definition at line 148 of file game.c.

4.29.3.3 getMaxPuntos() `unsigned long long getMaxPuntos ( void )`

Devuelve el puntaje máximo del jugador.

Returns

unsigned long long

Definition at line 203 of file game.c.

4.29.3.4 getNivel() `int getNivel ( void )`

Devuelve el nivel//run del jugador.

Returns

int

Definition at line 208 of file game.c.

4.29.3.5 getNombre() `char * getNombre ( void )`

Devuelve el nombre del jugador.

Returns

char*

Definition at line 194 of file game.c.

4.29.3.6 getPuntos() `unsigned long long getPuntos ( void )`

Devuelve el puntaje del jugador.

Returns

unsigned long long

Definition at line 198 of file game.c.

4.29.3.7 inicializarJuego() `void inicializarJuego ( void )`

Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.

Definition at line 314 of file game.c.

4.29.3.8 limpiarMapa() `void limpiarMapa ( )`

Definition at line 116 of file game.c.

4.29.3.9 moverAdelante() `void moverAdelante ( void )`

Avanza el jugador.

Definition at line 245 of file game.c.

4.29.3.10 moverAtras() `void moverAtras ( void )`

Retrocede el jugador.

Definition at line 277 of file game.c.

4.29.3.11 moverCarrriles() `void moverCarrriles ( int x )`

Definition at line 120 of file game.c.

4.29.3.12 moverDcha() `void moverDcha ( void )`

Mueve el jugador a la derecha.

Definition at line 293 of file game.c.

4.29.3.13 moverIzda() `void moverIzda (`

`void )`

Mueve el jugador a la izquierda.

Definition at line 283 of file game.c.

4.29.3.14 pausarJuego() `void pausarJuego (`

`void )`

Pausa el juego.

Definition at line 335 of file game.c.

4.29.3.15 perderVida() `void perderVida (`

`void )`

Resta una vida.

Definition at line 303 of file game.c.

4.29.3.16 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 376 of file game.c.

4.29.3.17 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 167 of file game.c.

4.29.3.18 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 213 of file game.c.

4.29.3.19 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 222 of file game.c.

4.29.3.20 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 189 of file game.c.

4.29.3.21 setMaxPuntos() `void setMaxPuntos ( unsigned long long max )`

Setea los puntos maximos del jugador.

Parameters

max	

Definition at line 111 of file game.c.

4.29.3.22 setNombre() `void setNombre ( char * nombre )`

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 106 of file game.c.

4.29.3.23 spawnearAutos() `void spawnearAutos ( )`

Definition at line 126 of file game.c.

4.29.3.24 tiempoRefrescoEntidades() `bool tiempoRefrescoEntidades ( void )`

Chequea si es tiempo de refrescar entidades según la plataforma.

Returns

true

false

Definition at line 184 of file game.c.

4.29.4 Variable Documentation

4.29.4.1 agua `bool agua`

Definition at line 51 of file game.c.

4.29.4.2 completo `uint32_t completo`

Definition at line 65 of file game.c.

4.29.4.3 derecho `uint16_t derecho`

Definition at line 68 of file game.c.

4.29.4.4 dificultad `int dificultad`

Definition at line 40 of file game.c.

4.29.4.5 disp_matriz `matriz_t disp_matriz [extern]`

Definition at line 37 of file display.c.

4.29.4.6 izquierdo `uint16_t izquierdo`

Definition at line 69 of file game.c.

4.29.4.7 jugador_1 `uint16_t jugador_1`

Definition at line 46 of file game.c.

4.29.4.8 jugador_2 `uint16_t jugador_2`

Definition at line 47 of file game.c.

4.29.4.9 jugador_posicion_oeste `int jugador_posicion_oeste`

Definition at line 43 of file game.c.

4.29.4.10 jugador_posicion_sur `int jugador_posicion_sur`

Definition at line 42 of file game.c.

4.29.4.11 jugando `bool jugando`

Definition at line 50 of file game.c.

4.29.4.12 mapa `matriz_t mapa`

Definition at line 62 of file game.c.

4.29.4.13 max_puntos `unsigned long long max_puntos`

Definition at line 49 of file game.c.

4.29.4.14 niv_actual `int niv_actual`

Definition at line 41 of file game.c.

4.29.4.15 nombre_jugador `char nombre_jugador[L_MAX]`

Definition at line 39 of file game.c.

4.29.4.16 pre_timeout `bool pre_timeout`

Definition at line 54 of file game.c.

4.29.4.17 puntos `unsigned long long puntos`

Definition at line 48 of file game.c.

4.29.4.18 ranas `uint16_t ranas`

Definition at line 45 of file game.c.

4.29.4.19 refresco_autos `bool refresco_autos`

Definition at line 53 of file game.c.

4.29.4.20 refresco_jugador `bool refresco_jugador`

Definition at line 52 of file game.c.

4.29.4.21 tiempo `clock_t tiempo`

Definition at line 56 of file game.c.

4.29.4.22 tiempo_alerta `clock_t tiempo_alerta`

Definition at line 61 of file game.c.

4.29.4.23 tiempo_inicio `clock_t tiempo_inicio`

Definition at line 57 of file game.c.

4.29.4.24 tiempo_referencia `clock_t tiempo_referencia`

Definition at line 58 of file game.c.

4.29.4.25 tiempo_refresco_autos `clock_t tiempo_refresco_autos`

Definition at line 60 of file game.c.

4.29.4.26 tiempo_refresco_jugador `clock_t tiempo_refresco_jugador`

Definition at line 59 of file game.c.

4.29.4.27 timeout `bool timeout`

Definition at line 55 of file game.c.

4.29.4.28 vidas `uint16_t vidas`

Definition at line 44 of file game.c.

4.30 game.c

Go to the documentation of this file.

```
00001
00008 /*****************************************************************************
00009  * INCLUDE HEADER FILES
00010  *****************************************************************************/
00011
00012 #include "../../game.h"
00013
00014 #include "bitmap.h"
00015 #include "../../display.h"
00016 #include "../../sound.h"
00017 #include "../../queue.h"
00018
00019 #include <string.h>
00020 #include <stdlib.h>
00021 #include <pthread.h>
00022 #include <time.h>
00023
00024 /*****************************************************************************
00025  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00026  *****************************************************************************/
00027
00028 #define POS_AUTOS_INICIO 4
00029 #define POS_AUTOS_FIN 13
00030 #define CANT_CARRILES 5
00031 #define L_MAX 64
00032
00033 /*****************************************************************************
00034  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00035  *****************************************************************************/
00036
00037 static struct
00038 {
```

```
00039     char nombre_jugador[L_MAX];
00040     int dificultad;
00041     int niv_actual;
00042     int jugador_posicion_sur;
00043     int jugador_posicion_oeste;
00044     uint16_t vidas;
00045     uint16_t ranas;
00046     uint16_t jugador_1;
00047     uint16_t jugador_2;
00048     unsigned long long puntos;
00049     unsigned long long max_puntos;
00050     bool jugando;
00051     bool agua;
00052     bool refresco_jugador;
00053     bool refresco_autos;
00054     bool pre_timeout; //para avisar si queda poco tiempo
00055     bool timeout;
00056     clock_t tiempo;
00057     clock_t tiempo_inicio;
00058     clock_t tiempo_referencia;
00059     clock_t tiempo_refresco_jugador;
00060     clock_t tiempo_refresco_autos;
00061     clock_t tiempo_alerta;
00062     matriz_t mapa;
00063     union
00064     {
00065         uint32_t completo;
00066         struct
00067         {
00068             uint16_t derecho;
00069             uint16_t izquierdo;
00070         };
00071     } carril[CANT_CARRILES];
00072 } juego;
00073
00074 /*****************************************************************************
00075  * VARIABLES WITH GLOBAL SCOPE
00076  *****************************************************************************/
00077
00078 extern matriz_t disp_matriz;
00079
00080 /*****************************************************************************
00081  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00082  *****************************************************************************/
00083
00084 static void reiniciarTimer();
00085
00086 /*****************************************************************************
00087  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00088  *****************************************************************************/
00089
00090 // +ej: static const int temperaturas_medias[4] = {23, 26, 24, 29};+
00091
00092 /*****************************************************************************
00093  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00094  *****************************************************************************/
00095
00096 static pthread_t ttiempo;
00097
00098 static void *threadTiempo(void *ptr);
00099
00100 /*****************************************************************************
00101  *****************************************************************************
00102                         GLOBAL FUNCTION DEFINITIONS
00103  *****************************************************************************
00104  *****************************************************************************/
00105
00106 void setNombre(char *nombre)
00107 {
00108     strcpy(juego.nombre_jugador, nombre);
00109 }
00110
00111 void setMaxPuntos(unsigned long long max)
00112 {
00113     juego.max_puntos = max;
00114 }
00115
00116 void limpiarMapa()
00117 {
00118 }
00119
00120 void moverCarrriles(int x)
00121 {
00122     for (int i = 0; i < 5; i++)
00123         juego.carril[i].completo <<= x;
00124 }
00125
```

```
00126 void spawnearAutos()
00127 {
00128     int i;
00129     for (i = 0; i < 5; i++)
00130     {
00131         if (juego.agua)
00132         {
00133             if (!(juego.carril[i].completo & 0b1111111111) && !(rand() % 10))
00134                 juego.carril[i].completo |= 0b111111;
00135             else if (!(juego.carril[i].completo & 0b1111111111111) && !(rand() % 20))
00136                 juego.carril[i].completo |= 0b11111111;
00137         }
00138         else
00139         {
00140             if (!(juego.carril[i].completo & 0b111111) && !(rand() % 10))
00141                 juego.carril[i].completo |= 0b11;
00142             else if (!(juego.carril[i].completo & 0b11111111) && !(rand() % 20))
00143                 juego.carril[i].completo |= 0b1111;
00144         }
00145     }
00146 }
00147
00148 void actualizarMapa()
00149 {
00150     if (juego.agua)
00151     {
00152         juego.mapa[2] = juego.ranas;
00153         juego.mapa[3] = juego.ranas;
00154     }
00155     else
00156     {
00157         juego.mapa[2] = 0;
00158         juego.mapa[3] = 0;
00159     }
00160     for (int i = 0; i < 5; i++)
00161     {
00162         juego.mapa[POS_AUTOS_INICIO + 2 * i] = juego.carril[i].izquierdo;
00163         juego.mapa[POS_AUTOS_INICIO + 2 * i + 1] = juego.carril[i].izquierdo;
00164     }
00165 }
00166
00167 void refrescar()
00168 {
00169     if (juego.refresco_autos)
00170     {
00171         moverCarrriles(1);
00172         spawnearAutos();
00173         juego.refresco_autos = false;
00174     }
00175     if (juego.refresco_jugador)
00176     {
00177         uint16_t tmp = juego.jugador_1;
00178         juego.jugador_1 = juego.jugador_2;
00179         juego.jugador_2 = tmp;
00180         juego.refresco_jugador = false;
00181     }
00182 }
00183
00184 bool tiempoRefrescoEntidades(void)
00185 {
00186     return juego.refresco_jugador || juego.refresco_autos;
00187 }
00188
00189 void setDificultad(int dificultad)
00190 {
00191     juego.dificultad = dificultad;
00192 }
00193
00194 char *getNombre()
00195 {
00196     return juego.nombre_jugador;
00197 }
00198 unsigned long long getPuntos()
00199 {
00200     return juego.puntos;
00201 }
00202
00203 unsigned long long getMaxPuntos()
00204 {
00205     return juego.max_puntos;
00206 }
00207
00208 int getNivel()
00209 {
00210     return juego.niv_actual;
00211 }
00212
```

```
00213 void reiniciarNivel()
00214 {
00215     juego.ranas = 0b1001001001001001;
00216     juego.agua = false;
00217     respawn();
00218     reiniciarTimer();
00219     reanudarJuego();
00220 }
00221
00222 void respawn()
00223 {
00224     juego.jugador_posicion_sur = CANT_FILAS - 1;
00225     if (!juego.agua)
00226     {
00227         juego.jugador_1 = 0b0000000100000000;
00228         juego.jugador_2 = 0b0000000010000000;
00229         juego.jugador_posicion_oeste = 7;
00230     }
00231
00232     for (int i = 0; i < CANT_CARRILES; i++)
00233         juego.carril[i].completo = 0;
00234     limpiarMatriz(juego.mapa);
00235
00236     for (int i = 0; i < 20; i++)
00237     {
00238         moverCarrriles(4);
00239         spawnearAutos();
00240     }
00241
00242     actualizarMapa();
00243 }
00244
00245 void moverAdelante()
00246 {
00247     if (juego.jugador_posicion_sur > 3)
00248         juego.jugador_posicion_sur--;
00249     else
00250     {
00251         if (!juego.agua)
00252         {
00253             juego.agua = true;
00254             respawn();
00255         }
00256         else
00257         {
00258             juego.puntos += 500;
00259             juego.ranas |= juego.jugador_1 | juego.jugador_2;
00260             if (juego.ranas == 0b1111111111111111)
00261             {
00262                 juego.puntos += 1000;
00263                 pausarJuego();
00264                 juego.niv_actual++;
00265                 reproducirEfecto(EFECTO_NIVEL_COMPLETO);
00266                 reiniciarNivel();
00267             }
00268             else
00269             {
00270                 juego.agua = false;
00271                 respawn();
00272             }
00273         }
00274     }
00275 }
00276
00277 void moverAtras()
00278 {
00279     if (juego.jugador_posicion_sur < 15)
00280         juego.jugador_posicion_sur++;
00281 }
00282
00283 void moverIzda()
00284 {
00285     if (juego.jugador_posicion_oeste > 0)
00286     {
00287         juego.jugador_posicion_oeste--;
00288         juego.jugador_1 «= 1;
00289         juego.jugador_2 «= 1;
00290     }
00291 }
00292
00293 void moverDcha()
00294 {
00295     if (juego.jugador_posicion_oeste < 14)
00296     {
00297         juego.jugador_posicion_oeste++;
00298         juego.jugador_1 »= 1;
00299         juego.jugador_2 »= 1;
```

```
00300     }
00301 }
00302
00303 void perderVida()
00304 {
00305     juego.agua ? reproducirEfecto(EFECTO_AHOGADO) : reproducirEfecto(EFECTO_IMPACTO);
00306     juego.agua = false;
00307     juego.vidas «= 1;
00308     if (!juego.vidas)
00309         queueInsertar(GAME_OVER);
00310     else
00311         respawn();
00312 }
00313
00314 void inicializarJuego()
00315 {
00316     juego.puntos = 0;
00317     juego.niv_actual = 1;
00318     juego.vidas = 0b1110000000000000;
00319 }
00320
00321 void reiniciarTimer()
00322 {
00323     juego.tiempo_inicio = CLOCKS_PER_SEC * (100 - (2 * (juego.dificultad + juego.niv_actual)));
00324     juego.tiempo = juego.tiempo_inicio;
00325     juego.tiempo_referencia = juego.tiempo_inicio;
00326     juego.tiempo_refresco_autos = CLOCKS_PER_SEC * (1 - 0.25 * (2 * (juego.dificultad +
      juego.niv_actual - 1)));
00327     juego.tiempo_refresco_jugador = CLOCKS_PER_SEC » 1;
00328     juego.tiempo_alerta = CLOCKS_PER_SEC * 15;
00329     juego.refresco_autos = false;
00330     juego.refresco_jugador = false;
00331     juego.timeout = false;
00332     juego.pre_timeout = false;
00333 }
00334
00335 void pausarJuego()
00336 {
00337     juego.jugando = false;
00338     pthread_join(ttiempo, NULL);
00339 }
00340
00341 void actualizarInterfaz()
00342 {
00343     actualizarMapa();
00344     copiarMatriz(disp_matriz, juego.mapa);
00345     disp_matriz[0] = juego.vidas;
00346
00347     // algoritmo para convertir el tiempo al indicador de 16 bits por aproximaciones sucesivas
00348     clock_t acc = 0, frac = juego.tiempo_inicio » 1;
00349     uint16_t tiempo_bits = 0;
00350     int bit_de_referencia = 0b0000000000000001;
00351     int j;
00352     for (j = 8; j; j »= 1, frac »= 1)
00353     {
00354         if (juego.tiempo > (acc + frac))
00355         {
00356             acc += frac;
00357             bit_de_referencia «= j;
00358             tiempo_bits |= bit_de_referencia - 1; // hago 1 todos los bits menos significativos que el
      de referencia
00359         }
00360     }
00361
00362     disp_matriz[1] = tiempo_bits;
00363
00364     disp_matriz[(juego.jugador_posicion_sur) - 1] |= juego.jugador_1;
00365     disp_matriz[juego.jugador_posicion_sur] |= juego.jugador_2;
00366
00367     actualizarDisplay();
00368
00369     if ((juego.mapa[(juego.jugador_posicion_sur) - 1]) & juego.jugador_1 ||
     (juego.mapa[(juego.jugador_posicion_sur)]) & juego.jugador_2)
00370         perderVida();
00371
00372     if (juego.timeout)
00373         queueInsertar(GAME_OVER);
00374 }
00375
00376 void reanudarJuego(void)
00377 {
00378     juego.jugando = true;
00379     pthread_create(&ttiempo, NULL, threadTiempo, NULL);
00380 }
00381
00382 /*****************************************************************************
00383  *****************************************************************************
```

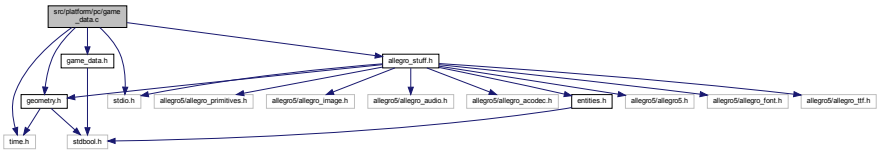
```
00384                         LOCAL FUNCTION DEFINITIONS
00385  ********************************************************************************
00386  ********************************************************************************/
00387
00388 static void *threadTiempo(void *ptr)
00389 {
00390     clock_t ref = clock();
00391     clock_t ref_autos = ref + juego.tiempo_refresco_autos, ref_jugador = ref +
      juego.tiempo_refresco_jugador;
00392     while (juego.jugando)
00393     {
00394         if (!(juego.timeout))
00395         {
00396             clock_t tiempo = clock();
00397             juego.tiempo = juego.tiempo_referencia - (tiempo - ref);
00398             juego.timeout = juego.tiempo <= 0;
00399             if (!juego.pre_timeout && juego.tiempo <= juego.tiempo_alerta)
00400             {
00401                 juego.pre_timeout = true;
00402                 reproducirEfecto(EFECTO_POCO_TIEMPO);
00403             }
00404             if (tiempo > ref_autos)
00405             {
00406                 juego.refresco_autos = true;
00407                 ref_autos = tiempo + juego.tiempo_refresco_autos;
00408             }
00409             if (tiempo > ref_jugador)
00410             {
00411                 juego.refresco_jugador = true;
00412                 ref_jugador = tiempo + juego.tiempo_refresco_jugador;
00413             }
00414         }
00415         else
00416             ref = clock();
00417     }
00418
00419     juego.tiempo_referencia = juego.tiempo;
00420
00421     return NULL;
00422 }
```

4.31 src/platform/pc/game_data.c File Reference

Source del modulo game_data. Inicializa, actualiza y muestra los datos del juego en PC.

```
#include "game_data.h"
#include <stdio.h>
#include <time.h>
#include "geometry.h"
#include "allegro_stuff.h"
```

Include dependency graph for game_data.c:



Data Structures

- struct data_t

Macros

- #define MAX_NAME_CHAR 20
- #define MAX_LIVES 3
- #define SCORE_PER_GOAL 500
- #define SCORE_PER_GOAL_COIN 750
- #define SCORE_PER_RUN 1000
- #define INITIAL_RUN_TIME_LEFT 30
- #define EXTRA_TIME_PER_GOAL 10
- #define EXTRA_TIME_PER_BONUS_GOAL 15
- #define TIME_LEFT_WARNING 10
- #define HUD_EXTRA_INFO_TIMING 120
- #define HUD_EXTRA_INFO_RATE 1

Enumerations

- enum **DATA_FLAGS** { **DATA_FLAG_STARTING** , **DATA_FLAG_NEXT_RUN** , **DATA_FLAG_TIME_↩ EXCEEDED** , **DATA_FLAG_GAME_OVER** }
- enum **HUD_EXTRAS** { **HUD_EXTRA_TIME** , **HUD_EXTRA_SCORE** , **HUD_EXTRA_RUN** , **HUD_↩ EXTRAS_MAX** }

Functions

- void game_data_init (void)

  Inicializa datos internos del juego.
- void game_data_update (void)

  Actualiza datos internos del juego.
- void game_data_draw (void)

  Grafica datos del juego (HUD pricipalmente)
- int game_data_get_lives (void)

  Devuelve vidas.
- void game_data_subtract_live (void)

  Resta una vida.
- unsigned long long game_data_get_score (void)

  Devuelve score.
- void game_data_add_score (void)

  Agrega score por llegar a la meta.
- void game_data_add_score_bonus (void)

  Agrega score por llegar a la meta con bonus (coins)
- void game_data_set_score_max (unsigned long long score)

  Carga el score maximo del jugador actual.
- unsigned long long game_data_get_score_max (void)

  Devuelve el score maximo del jugador actual.
- int game_data_get_run_number (void)

  Devuelve el numero de run.
- void game_data_next_run (void)

  Indica que se pase a la siguiente run.
- int game_data_get_run_time_left (void)

  Devuelve el tiempo restante de la run en segundos.
- void game_data_add_run_time_goal (void)

  Agrega tiempo a la run por llegar a una meta.

- void game_data_add_run_time_goal_bonus (void)
  Agrega tiempo (más) a la run por llegar a una meta con coin.
- unsigned long game_data_get_frames (void)
  Devuelve los frames transcurridos del juego.
- int game_data_get_timer_in_sec (void)
  Devuelve el tiempo transcurrido en segundos.
- void game_data_set_diff (int diff)
  Setea dificultad.
- int game_data_get_diff (void)
  Devuelve dificultad.
- void game_data_clear_name (void)
  Limpia el nombre del jugador.
- void game_data_overwrite_name (char ∗name)
  Sobreescribe el nombre del jugador.
- void game_data_add_name_letter (char letter)
  Agrega una letra la nombre del jugador.
- char ∗ game_data_get_name (void)
  Devuelve puntero al nombre del jugador.
- bool game_data_get_goal_state (unsigned int goal)
  Revisa si un punto de llegada es valido o no (vacio o lleno)
- void game_data_set_goal (unsigned int goal)
  Setea un goal como completado.
- void game_data_reset_goals (void)
  Habilita todos los goals.
- bool game_data_get_game_over_flag (void)
  Devuelve flag de game over.
- bool game_data_are_goals_full (void)
  Avisa si estan todas las metas completas.
- unsigned long long game_data_get_old_max_score (void)
  Devuelve el score maximo sin actualizar al terminar el juego.

4.31.1 Detailed Description

Source del modulo game_data. Inicializa, actualiza y muestra los datos del juego en PC.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file game_data.c.

4.31.2 Macro Definition Documentation

4.31.2.1 EXTRA_TIME_PER_BONUS_GOAL `#define EXTRA_TIME_PER_BONUS_GOAL 15`

Definition at line 37 of file game_data.c.

4.31.2.2 EXTRA_TIME_PER_GOAL `#define EXTRA_TIME_PER_GOAL 10`

Definition at line 36 of file game_data.c.

4.31.2.3 HUD_EXTRA_INFO_RATE `#define HUD_EXTRA_INFO_RATE 1`

Definition at line 42 of file game_data.c.

4.31.2.4 HUD_EXTRA_INFO_TIMING `#define HUD_EXTRA_INFO_TIMING 120`

Definition at line 41 of file game_data.c.

4.31.2.5 INITIAL_RUN_TIME_LEFT `#define INITIAL_RUN_TIME_LEFT 30`

Definition at line 34 of file game_data.c.

4.31.2.6 MAX_LIVES `#define MAX_LIVES 3`

Definition at line 28 of file game_data.c.

4.31.2.7 MAX_NAME_CHAR `#define MAX_NAME_CHAR 20`

Definition at line 26 of file game_data.c.

4.31.2.8 SCORE_PER_GOAL `#define SCORE_PER_GOAL 500`

Definition at line 30 of file game_data.c.

4.31.2.9 SCORE_PER_GOAL_COIN `#define SCORE_PER_GOAL_COIN 750`

Definition at line 31 of file game_data.c.

4.31.2.10 SCORE_PER_RUN `#define SCORE_PER_RUN 1000`

Definition at line 32 of file game_data.c.

4.31.2.11 TIME_LEFT_WARNING `#define TIME_LEFT_WARNING 10`

Definition at line 39 of file game_data.c.

4.31.3 Enumeration Type Documentation

4.31.3.1 DATA_FLAGS `enum DATA_FLAGS`

Definition at line 75 of file game_data.c.

4.31.3.2 HUD_EXTRAS `enum HUD_EXTRAS`

Definition at line 84 of file game_data.c.

4.31.4 Function Documentation

4.31.4.1 game_data_add_name_letter() `void game_data_add_name_letter ( char letter )`

Agrega una letra la nombre del jugador.

Parameters

letter	Letra

Definition at line 371 of file game_data.c.

4.31.4.2 game_data_add_run_time_goal() `void game_data_add_run_time_goal ( void )`

Agrega tiempo a la run por llegar a una meta.

Definition at line 329 of file game_data.c.

4.31.4.3 game_data_add_run_time_goal_bonus() `void game_data_add_run_time_goal_bonus ( void )`

Agrega tiempo (más) a la run por llegar a una meta con coin.

Definition at line 335 of file game_data.c.

4.31.4.4 game_data_add_score() `void game_data_add_score ( void )`

Agrega score por llegar a la meta.

Definition at line 292 of file game_data.c.

4.31.4.5 game_data_add_score_bonus() `void game_data_add_score_bonus ( void )`

Agrega score por llegar a la meta con bonus (coins)

Definition at line 298 of file game_data.c.

4.31.4.6 game_data_are_goals_full() `bool game_data_are_goals_full ( void )`

Avisa si estan todas las metas completas.

Returns

true Si

false No

Definition at line 418 of file game_data.c.

4.31.4.7 game_data_clear_name() `void game_data_clear_name (`

`void )`

Limpia el nombre del jugador.

Definition at line 361 of file game_data.c.

4.31.4.8 game_data_draw() `void game_data_draw (`

`void )`

Grafica datos del juego (HUD pricipalmente)

Definition at line 271 of file game_data.c.

4.31.4.9 game_data_get_diff() `int game_data_get_diff (`

`void )`

Devuelve dificultad.

Returns

int

Definition at line 356 of file game_data.c.

4.31.4.10 game_data_get_frames() `unsigned long game_data_get_frames (`

`void )`

Devuelve los frames transcurridos del juego.

Returns

unsigned long Frames transcurridos

Definition at line 341 of file game_data.c.

4.31.4.11 game_data_get_game_over_flag() `bool game_data_get_game_over_flag (`

`void )`

Devuelve flag de game over.

Returns

true Game over

false No game over

Definition at line 410 of file game_data.c.

4.31.4.12 game_data_get_goal_state() `bool game_data_get_goal_state (`

`unsigned int goal )`

Revisa si un punto de llegada es valido o no (vacio o lleno)

Parameters

goal	0 a MAX_GOALS-1

Returns

true Invalido

false Valido

Definition at line 393 of file game_data.c.

4.31.4.13 game_data_get_lives() `int game_data_get_lives ( void )`

Devuelve vidas.

Returns

int vidas

Definition at line 277 of file game_data.c.

4.31.4.14 game_data_get_name() `char * game_data_get_name ( void )`

Devuelve puntero al nombre del jugador.

Returns

char*

Definition at line 388 of file game_data.c.

4.31.4.15 game_data_get_old_max_score() `unsigned long long game_data_get_old_max_score ( void )`

Devuelve el score maximo sin actualizar al terminar el juego.

Returns

unsigned long long

Definition at line 436 of file game_data.c.

4.31.4.16 game_data_get_run_number() int game_data_get_run_number (void)

Devuelve el numero de run.

Returns

int Numero de run

Definition at line 314 of file game_data.c.

4.31.4.17 game_data_get_run_time_left() int game_data_get_run_time_left (void)

Devuelve el tiempo restante de la run en segundos.

Returns

int Tiempo restante

Definition at line 324 of file game_data.c.

4.31.4.18 game_data_get_score() unsigned long long game_data_get_score (void)

Devuelve score.

Returns

int

Definition at line 287 of file game_data.c.

4.31.4.19 game_data_get_score_max() unsigned long long game_data_get_score_max (void)

Devuelve el score maximo del jugador actual.

Returns

unsigned long long Score maximo

Definition at line 309 of file game_data.c.

4.31.4.20 game_data_get_timer_in_sec() int game_data_get_timer_in_sec (void)

Devuelve el tiempo transcurrido en segundos.

Returns

int Segundos transcurridos

Definition at line 346 of file game_data.c.

4.31.4.21 game_data_init() void game_data_init (void)

Inicializa datos internos del juego.

Definition at line 209 of file game_data.c.

4.31.4.22 game_data_next_run() void game_data_next_run (void)

Indica que se pase a la siguiente run.

Definition at line 319 of file game_data.c.

4.31.4.23 game_data_overwrite_name() void game_data_overwrite_name (char * *name*)

Sobreescribe el nombre del jugador.

Parameters

name	

Definition at line 366 of file game_data.c.

4.31.4.24 game_data_reset_goals() void game_data_reset_goals (void)

Habilita todos los goals.

Definition at line 403 of file game_data.c.

4.31.4.25 game_data_set_diff() `void game_data_set_diff ( int diff )`

Setea dificultad.

Parameters

diff	enum DIFFICULTIES

Definition at line 351 of file game_data.c.

4.31.4.26 game_data_set_goal() `void game_data_set_goal ( unsigned int goal )`

Setea un goal como completado.

Parameters

goal	0 a MAX_GOALS-1

Definition at line 398 of file game_data.c.

4.31.4.27 game_data_set_score_max() `void game_data_set_score_max ( unsigned long long score )`

Carga el score maximo del jugador actual.

Parameters

score	

Definition at line 304 of file game_data.c.

4.31.4.28 game_data_subtract_live() `void game_data_subtract_live ( void )`

Resta una vida.

Definition at line 282 of file game_data.c.

4.31.4.29 game_data_update() void game_data_update (void)

Actualiza datos internos del juego.

Definition at line 227 of file game_data.c.

4.31.5 Variable Documentation

4.31.5.1 flag bool flag

Definition at line 95 of file game_data.c.

4.31.5.2 shifter int shifter

Definition at line 98 of file game_data.c.

4.31.5.3 timer int timer

Definition at line 97 of file game_data.c.

4.31.5.4 value int value

Definition at line 96 of file game_data.c.

4.32 game_data.c

Go to the documentation of this file.

```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "game_data.h"
00017 #include <stdio.h>
00018 #include <time.h>
00019 #include "geometry.h"
00020 #include "allegro_stuff.h"
00021
00022 /*******************************************************************************
00023  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00024  ******************************************************************************/
00025
00026 #define MAX_NAME_CHAR 20
00027
00028 #define MAX_LIVES 3
00029
00030 #define SCORE_PER_GOAL 500      // puntaje por llegar a la meta
00031 #define SCORE_PER_GOAL_COIN 750 // puntaje por llegar a la meta con coin
```

```
00032 #define SCORE_PER_RUN 1000      // puntaje por completar una run
00033
00034 #define INITIAL_RUN_TIME_LEFT 30 //tiempo inicial de una partida
00035
00036 #define EXTRA_TIME_PER_GOAL 10       // 10s extras por llegar a una meta
00037 #define EXTRA_TIME_PER_BONUS_GOAL 15 // 15s extras por llegar a una meta con coin
00038
00039 #define TIME_LEFT_WARNING 10 // warning 10s antes del timeout
00040
00041 #define HUD_EXTRA_INFO_TIMING 120 // frames que duran los mensajes emergentes
00042 #define HUD_EXTRA_INFO_RATE 1     // "velocidad" de desplazamiento de mensajes
00043
00044 /*****************************************************************************
00045  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00046  *****************************************************************************/
00047
00048 typedef struct
00049 {
00050     int lives;
00051     unsigned long long score;
00052     unsigned long long score_max;
00053
00054     struct
00055     {
00056         int number;    // numero de run actual
00057         int time_left; // tiempo restante en la run
00058         int time;      // tiempo de la run actual
00059         long time_ref; // referencia de tiempo global de la run
00060     } run;
00061
00062     unsigned long frames;
00063     int timer_in_sec;
00064
00065     int difficulty;
00066
00067     char name[MAX_NAME_CHAR];
00068
00069     unsigned char flag;
00070
00071     bool goals[MAX_GOALS];
00072
00073 } data_t;
00074
00075 enum DATA_FLAGS
00076 {
00077     DATA_FLAG_STARTING,
00078     DATA_FLAG_NEXT_RUN,
00079     DATA_FLAG_TIME_EXCEEDED,
00080     DATA_FLAG_GAME_OVER
00081 };
00082
00083 // Tipos de mensajes emergentes en el HUD
00084 enum HUD_EXTRAS
00085 {
00086     HUD_EXTRA_TIME,
00087     HUD_EXTRA_SCORE,
00088     HUD_EXTRA_RUN,
00089     HUD_EXTRAS_MAX
00090 };
00091
00092 // Estructura para el manejo de mensajes emergentes en el HUD
00093 static struct
00094 {
00095     bool flag;   // Activado ó no
00096     int value;   // Valor extra a mostrar
00097     int timer;   // Contador interno para dejar de mostrar
00098     int shifter; // Contador interno para desplazar el mensaje verticalmente
00099 } hud_extra_stuff[HUD_EXTRAS_MAX];
00100
00101 /*****************************************************************************
00102  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00103  *****************************************************************************/
00104
00109 static void data_init(void);
00110
00115 static void data_update(void);
00116
00121 static void hud_draw(void);
00122
00127 static void draw_reached_goals(void);
00128
00133 static void next_run(void);
00134
00140 static void trigger_show_adding_time(int extra);
00141
00147 static void trigger_show_adding_score(int extra);
00148
```

```
00153 static void trigger_show_adding_run(void);
00154
00159 static void draw_extra_time(void);
00160
00165 static void draw_extra_score(void);
00166
00171 static void draw_extra_run(void);
00172
00173 /*********************************************************************************
00174  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00175  *********************************************************************************/
00176
00177 // Datos internos del juego
00178 static data_t data;
00179
00180 // Referencia temporal del inicio del juego
00181 static long time_ref;
00182
00183 static int char_h; // altura de un caracter
00184 static int char_w; // ancho de un caracter
00185
00186 static ALLEGRO_COLOR text_color;
00187
00188 // Flag para triggear sonido de poco tiempo
00189 static bool flag_low_time_warning;
00190
00191 // Auxiliar para hacer acciones en los cambios de segundos
00192 static int last_loop_time;
00193
00194 // Tiempo restante inicial de una nueva run. Se puede modificar externamente
00195 static int new_run_time_left;
00196
00197 // Auxiliar para mostrar el score gradualmente en el HUD
00198 static unsigned long long score_display;
00199
00200 // Score maximo no actualizado en game over
00201 static unsigned long long max_score_no_updated;
00202
00203 /*********************************************************************************
00204  *********************************************************************************
00205             GLOBAL FUNCTION DEFINITIONS
00206  *********************************************************************************
00207  *********************************************************************************/
00208
00209 void game_data_init(void)
00210 {
00211     time_ref = time(NULL);
00212
00213     char_h = allegro_get_var_font_h();
00214     char_w = allegro_get_var_font_w();
00215
00216     text_color = al_map_rgb(255, 255, 255);
00217
00218     flag_low_time_warning = false;
00219
00220     new_run_time_left = INITIAL_RUN_TIME_LEFT;
00221
00222     score_display = 0;
00223
00224     data_init();
00225 }
00226
00227 void game_data_update(void)
00228 {
00229     data_update();
00230
00231     if (data.flag == DATA_FLAG_NEXT_RUN)
00232     {
00233         next_run();
00234         data.flag = DATA_FLAG_STARTING;
00235     }
00236
00237     if (data.run.time_left == TIME_LEFT_WARNING && !flag_low_time_warning)
00238     {
00239         allegro_sound_play_effect_low_time();
00240
00241         flag_low_time_warning = true;
00242     }
00243     else if (data.run.time_left < TIME_LEFT_WARNING)
00244         flag_low_time_warning = false;
00245
00246     if (max_score_no_updated != data.score_max)
00247         max_score_no_updated = data.score_max;
00248
00249     if (data.run.time_left == 0)
00250     {
00251         // data.flag = DATA_FLAG_TIME_EXCEEDED;
```

```
00252         game_data_subtract_live();
00253         allegro_sound_play_effect_no_time();
00254         data.flag = DATA_FLAG_GAME_OVER;
00255     }
00256 
00257     if (data.lives == 0)
00258     {
00259         data.flag = DATA_FLAG_GAME_OVER;
00260     }
00261 
00262     // Las primeras 2 runs se reduce el timer en 5 segundos.
00263     if (data.run.number <= 1)
00264         new_run_time_left = INITIAL_RUN_TIME_LEFT - (5 * (data.run.number + 1));
00265 
00266     // Despues se reduce de a 2 segundos por run hasta llegar a 10 segundos.
00267     else if (data.run.number <= 6)
00268         new_run_time_left = INITIAL_RUN_TIME_LEFT - (2 * (data.run.number + 1));
00269 }
00270 
00271 void game_data_draw(void)
00272 {
00273     hud_draw();
00274     draw_reached_goals();
00275 }
00276 
00277 int game_data_get_lives(void)
00278 {
00279     return (data.lives);
00280 }
00281 
00282 void game_data_subtract_live(void)
00283 {
00284     data.lives--;
00285 }
00286 
00287 unsigned long long game_data_get_score(void)
00288 {
00289     return (data.score);
00290 }
00291 
00292 void game_data_add_score(void)
00293 {
00294     data.score += SCORE_PER_GOAL;
00295     trigger_show_adding_score(SCORE_PER_GOAL);
00296 }
00297 
00298 void game_data_add_score_bonus(void)
00299 {
00300     data.score += SCORE_PER_GOAL_COIN;
00301     trigger_show_adding_score(SCORE_PER_GOAL_COIN);
00302 }
00303 
00304 void game_data_set_score_max(unsigned long long score)
00305 {
00306     data.score_max = score;
00307 }
00308 
00309 unsigned long long game_data_get_score_max(void)
00310 {
00311     return data.score_max;
00312 }
00313 
00314 int game_data_get_run_number(void)
00315 {
00316     return (data.run.number);
00317 }
00318 
00319 void game_data_next_run(void)
00320 {
00321     data.flag = DATA_FLAG_NEXT_RUN;
00322 }
00323 
00324 int game_data_get_run_time_left(void)
00325 {
00326     return (data.run.time_left);
00327 }
00328 
00329 void game_data_add_run_time_goal(void)
00330 {
00331     data.run.time_left += EXTRA_TIME_PER_GOAL;
00332     trigger_show_adding_time(EXTRA_TIME_PER_GOAL);
00333 }
00334 
00335 void game_data_add_run_time_goal_bonus(void)
00336 {
00337     data.run.time_left += EXTRA_TIME_PER_BONUS_GOAL;
00338     trigger_show_adding_time(EXTRA_TIME_PER_BONUS_GOAL);
```

```
00339 }
00340
00341 unsigned long game_data_get_frames(void)
00342 {
00343     return (data.frames);
00344 }
00345
00346 int game_data_get_timer_in_sec(void)
00347 {
00348     return (data.timer_in_sec);
00349 }
00350
00351 void game_data_set_diff(int diff)
00352 {
00353     data.difficulty = diff;
00354 }
00355
00356 int game_data_get_diff(void)
00357 {
00358     return (data.difficulty);
00359 }
00360
00361 void game_data_clear_name(void)
00362 {
00363     memset(data.name, 0, MAX_NAME_CHAR);
00364 }
00365
00366 void game_data_overwrite_name(char *name)
00367 {
00368     strcpy(data.name, name);
00369 }
00370
00371 void game_data_add_name_letter(char letter)
00372 {
00373     int length = strlen(data.name);
00374
00375     if ((letter == ALLEGRO_KEY_BACKSPACE) && (length > 0))
00376     {
00377         data.name[length - 1] = 0;
00378     }
00379
00380     else if (letter >= ALLEGRO_KEY_A && letter <= ALLEGRO_KEY_Z && length < MAX_NAME_CHAR)
00381     {
00382         letter += '@';
00383         data.name[length] = letter;
00384         data.name[length + 1] = 0;
00385     }
00386 }
00387
00388 char *game_data_get_name(void)
00389 {
00390     return (data.name);
00391 }
00392
00393 bool game_data_get_goal_state(unsigned int goal)
00394 {
00395     return data.goals[goal];
00396 }
00397
00398 void game_data_set_goal(unsigned int goal)
00399 {
00400     data.goals[goal] = true;
00401 }
00402
00403 void game_data_reset_goals(void)
00404 {
00405     int i;
00406     for (i = 0; i < MAX_GOALS; i++)
00407         data.goals[i] = false;
00408 }
00409
00410 bool game_data_get_game_over_flag(void)
00411 {
00412     if (data.flag == DATA_FLAG_GAME_OVER)
00413         return true;
00414     else
00415         return false;
00416 }
00417
00418 bool game_data_are_goals_full(void)
00419 {
00420     bool state = true;
00421
00422     int i;
00423     for (i = 0; i < MAX_GOALS; i++)
00424     {
00425         // si alguno esa vacio...
```

```
00426         if (!data.goals[i])
00427         {
00428             state = false;
00429             break;
00430         }
00431     }
00432
00433     return state;
00434 }
00435
00436 unsigned long long game_data_get_old_max_score(void)
00437 {
00438     return max_score_no_updated;
00439 }
00440
00441 /*****************************************************************************
00442  *****************************************************************************
00443             LOCAL FUNCTION DEFINITIONS
00444  *****************************************************************************
00445  *****************************************************************************/
00446
00447 static void data_init(void)
00448 {
00449     data.frames = 0;
00450     data.lives = MAX_LIVES;
00451     data.run.number = 0;
00452     data.run.time_left = new_run_time_left;
00453     data.run.time = 0;
00454     data.run.time_ref = time(NULL);
00455     data.score = 0;
00456     data.timer_in_sec = 0;
00457
00458     data.flag = DATA_FLAG_STARTING;
00459
00460     last_loop_time = 0;
00461
00462     game_data_reset_goals();
00463 }
00464
00465 static void data_update(void)
00466 {
00467
00468     // diferencia entre el tiempo actual y el de referencia
00469     data.timer_in_sec = time(NULL) - time_ref;
00470     data.frames++;
00471
00472     data.run.time = time(NULL) - data.run.time_ref;
00473
00474     if (data.run.time > last_loop_time)
00475     {
00476         data.run.time_left--;
00477         last_loop_time++;
00478     }
00479     last_loop_time = data.run.time;
00480 }
00481
00482 static void hud_draw(void)
00483 {
00484     // Dibuja la puntuacion en pantalla.
00485
00486     int i;
00487
00488     // Graduacion del score a mostrar para que vaya incrementando de apoco
00489     if (score_display != data.score)
00490     {
00491         int shifter;
00492
00493         for (i = 2, shifter = 0; i > 0; i--)
00494         {
00495             shifter = 1 « i;
00496             if (score_display <= (data.score - shifter))
00497                 score_display += shifter;
00498         }
00499     }
00500
00501     al_draw_textf(
00502         allegro_get_var_font(),
00503         text_color,
00504         1, 1, // Arriba a la izquierda.
00505         0,
00506         "Score: %06lld", // 6 cifras (por ahi es mucho).
00507         score_display);
00508
00509     draw_extra_score();
00510
00511     // Dibuja el numero de vuelta.
00512     al_draw_textf(
```

```
00513         allegro_get_var_font(),
00514         text_color,
00515         1, CELL_H - char_h - 5, // Para que quede abaj de la puntuacion en pantalla.
00516         0,
00517         "Run: %02d", // 2 cifras. No me acuerdo si esta bien asi.
00518         data.run.number);
00519
00520     draw_extra_run();
00521
00522     // Segundos.
00523     al_draw_textf(
00524         allegro_get_var_font(),
00525         text_color,
00526         al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w, 1,
00527         0,
00528         "Played Time: %04d",
00529         data.timer_in_sec);
00530
00531     // Tiempo restante
00532     al_draw_textf(
00533         allegro_get_var_font(),
00534         text_color,
00535         al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w,
00536         CELL_H - char_h - 5,
00537         0,
00538         "Time Left: %03d",
00539         data.run.time_left);
00540
00541     draw_extra_time();
00542
00543     // Dibuja vidas.
00544     for (int i = 0; i < data.lives; i++) // No se si la rana tiene 'frog.lives' pero aca va el
      equivalente.
00545         al_draw_bitmap(
00546             sprites.heart,
00547             DISPLAY_W - 100 + SPRITE_SIZE_HEART * (data.lives - i - 1),
00548             (CELL_H - char_h - 5) / 2,
00549             0);
00550 }
00551
00552 static void draw_reached_goals(void)
00553 {
00554     int i;
00555     for (i = 0; i < MAX_GOALS; i++)
00556     {
00557         // si algun goal fue alcanzado...
00558         if (data.goals[i])
00559             al_draw_bitmap(sprites.frog[6],
00560                            goal_cols[i] * CELL_W + FROG_OFFSET_X - 1,
00561                            CELL_H + FROG_OFFSET_Y + GOAL_ROW_OFFSET_Y_FIX,
00562                            0);
00563     }
00564 }
00565
00566 static void next_run(void)
00567 {
00568     data.run.number++;
00569     data.run.time_left = new_run_time_left;
00570     data.run.time = 0;
00571     data.run.time_ref = time(NULL);
00572
00573     data.score += SCORE_PER_RUN;
00574     trigger_show_adding_score(SCORE_PER_RUN);
00575
00576     last_loop_time = 0;
00577
00578     flag_low_time_warning = false;
00579
00580     trigger_show_adding_run();
00581
00582     game_data_reset_goals();
00583 }
00584
00585 static void trigger_show_adding_time(int extra)
00586 {
00587     hud_extra_stuff[HUD_EXTRA_TIME].flag = true;
00588     hud_extra_stuff[HUD_EXTRA_TIME].value = extra;
00589     hud_extra_stuff[HUD_EXTRA_TIME].timer = 1;
00590     hud_extra_stuff[HUD_EXTRA_TIME].shifter = 0;
00591 }
00592
00593 static void trigger_show_adding_score(int extra)
00594 {
00595     hud_extra_stuff[HUD_EXTRA_SCORE].flag = true;
00596     hud_extra_stuff[HUD_EXTRA_SCORE].value = extra;
00597     hud_extra_stuff[HUD_EXTRA_SCORE].timer = 1;
00598     hud_extra_stuff[HUD_EXTRA_SCORE].shifter = 0;
```

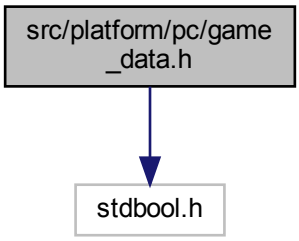
```
00599 }
00600
00601 static void trigger_show_adding_run()
00602 {
00603     hud_extra_stuff[HUD_EXTRA_RUN].flag = true;
00604     hud_extra_stuff[HUD_EXTRA_RUN].value = 1;
00605     hud_extra_stuff[HUD_EXTRA_RUN].timer = 1;
00606     hud_extra_stuff[HUD_EXTRA_RUN].shifter = 0;
00607 }
00608
00609 static void draw_extra_time(void)
00610 {
00611     if (hud_extra_stuff[HUD_EXTRA_TIME].flag)
00612     {
00613         al_draw_textf(
00614             allegro_get_var_font(),
00615             text_color,
00616             al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w +
00617                 al_get_text_width(allegro_get_var_font(), "Time left: "),
00618             CELL_H - char_h - 5 + hud_extra_stuff[HUD_EXTRA_TIME].shifter,
00619             0,
00620             "%+3d",
00621             hud_extra_stuff[HUD_EXTRA_TIME].value);
00622
00623         hud_extra_stuff[HUD_EXTRA_TIME].shifter += HUD_EXTRA_INFO_RATE;
00624
00625         if (!(hud_extra_stuff[HUD_EXTRA_TIME].timer++ % HUD_EXTRA_INFO_TIMING))
00626             hud_extra_stuff[HUD_EXTRA_TIME].flag = false;
00627     }
00628 }
00629
00630 static void draw_extra_score(void)
00631 {
00632     if (hud_extra_stuff[HUD_EXTRA_SCORE].flag)
00633     {
00634         al_draw_textf(
00635             allegro_get_var_font(),
00636             text_color,
00637             1 + al_get_text_width(allegro_get_var_font(), "Score: "),
00638             1 + hud_extra_stuff[HUD_EXTRA_SCORE].shifter,
00639             0,
00640             "%+6d",
00641             hud_extra_stuff[HUD_EXTRA_SCORE].value);
00642
00643         hud_extra_stuff[HUD_EXTRA_SCORE].shifter += HUD_EXTRA_INFO_RATE;
00644
00645         if (!(hud_extra_stuff[HUD_EXTRA_SCORE].timer++ % HUD_EXTRA_INFO_TIMING))
00646             hud_extra_stuff[HUD_EXTRA_SCORE].flag = false;
00647     }
00648 }
00649
00650 static void draw_extra_run(void)
00651 {
00652     if (hud_extra_stuff[HUD_EXTRA_RUN].flag)
00653     {
00654         al_draw_textf(
00655             allegro_get_var_font(),
00656             text_color,
00657             1 + al_get_text_width(allegro_get_var_font(), "Run: "),
00658             CELL_H - char_h - 5 + hud_extra_stuff[HUD_EXTRA_RUN].shifter,
00659             0,
00660             "%+2d",
00661             hud_extra_stuff[HUD_EXTRA_RUN].value);
00662
00663         hud_extra_stuff[HUD_EXTRA_RUN].shifter += HUD_EXTRA_INFO_RATE;
00664
00665         if (!(hud_extra_stuff[HUD_EXTRA_RUN].timer++ % HUD_EXTRA_INFO_TIMING))
00666             hud_extra_stuff[HUD_EXTRA_RUN].flag = false;
00667     }
00668 }
```

4.33 src/platform/pc/game_data.h File Reference

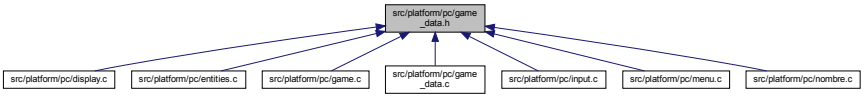
Header del modulo game_data. Prototipos de funciones globales que hacen al manejo de los datos del juego en PC:

```
#include <stdbool.h>
```

Include dependency graph for game_data.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum **DIFFICULTIES** { **DIFFICULTIES_EASY** = 1 , **DIFFICULTIES_NORMAL** , **DIFFICULTIES_HARD** }

Functions

- void game_data_init (void)

  Inicializa datos internos del juego.
- void game_data_update (void)

  Actualiza datos internos del juego.
- void game_data_draw (void)

  Grafica datos del juego (HUD pricipalmente)
- int game_data_get_lives (void)

  Devuelve vidas.
- void game_data_subtract_live (void)

  Resta una vida.
- unsigned long long game_data_get_score (void)

  Devuelve score.
- void game_data_add_score (void)

  Agrega score por llegar a la meta.
- void game_data_add_score_bonus (void)

  Agrega score por llegar a la meta con bonus (coins)
- void game_data_set_score_max (unsigned long long score)

  Carga el score maximo del jugador actual.

- unsigned long long game_data_get_score_max (void)
  Devuelve el score maximo del jugador actual.
- int game_data_get_run_number (void)
  Devuelve el numero de run.
- void game_data_next_run (void)
  Indica que se pase a la siguiente run.
- int game_data_get_run_time_left (void)
  Devuelve el tiempo restante de la run en segundos.
- void game_data_add_run_time_goal (void)
  Agrega tiempo a la run por llegar a una meta.
- void game_data_add_run_time_goal_bonus (void)
  Agrega tiempo (más) a la run por llegar a una meta con coin.
- unsigned long game_data_get_frames (void)
  Devuelve los frames transcurridos del juego.
- int game_data_get_timer_in_sec (void)
  Devuelve el tiempo transcurrido en segundos.
- void game_data_set_diff (int diff)
  Setea dificultad.
- int game_data_get_diff (void)
  Devuelve dificultad.
- void game_data_clear_name (void)
  Limpia el nombre del jugador.
- void game_data_overwrite_name (char *name)
  Sobreescribe el nombre del jugador.
- void game_data_add_name_letter (char letter)
  Agrega una letra la nombre del jugador.
- char * game_data_get_name (void)
  Devuelve puntero al nombre del jugador.
- bool game_data_get_goal_state (unsigned int goal)
  Revisa si un punto de llegada es valido o no (vacio o lleno)
- void game_data_set_goal (unsigned int goal)
  Setea un goal como completado.
- void game_data_reset_goals (void)
  Habilita todos los goals.
- bool game_data_get_game_over_flag (void)
  Devuelve flag de game over.
- bool game_data_are_goals_full (void)
  Avisa si estan todas las metas completas.
- unsigned long long game_data_get_old_max_score (void)
  Devuelve el score maximo sin actualizar al terminar el juego.

4.33.1 Detailed Description

Header del modulo game_data. Prototipos de funciones globales que hacen al manejo de los datos del juego en PC:

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file game_data.h.

4.33.2 Enumeration Type Documentation

4.33.2.1 DIFFICULTIES `enum DIFFICULTIES`

Definition at line 25 of file game_data.h.

4.33.3 Function Documentation

4.33.3.1 game_data_add_name_letter() `void game_data_add_name_letter ( char letter )`

Agrega una letra la nombre del jugador.

Parameters

letter	Letra

Definition at line 371 of file game_data.c.

4.33.3.2 game_data_add_run_time_goal() `void game_data_add_run_time_goal ( void )`

Agrega tiempo a la run por llegar a una meta.

Definition at line 329 of file game_data.c.

4.33.3.3 game_data_add_run_time_goal_bonus() `void game_data_add_run_time_goal_bonus ( void )`

Agrega tiempo (más) a la run por llegar a una meta con coin.

Definition at line 335 of file game_data.c.

4.33.3.4 game_data_add_score() `void game_data_add_score ( void )`

Agrega score por llegar a la meta.

Definition at line 292 of file game_data.c.

4.33.3.5 game_data_add_score_bonus() `void game_data_add_score_bonus ( void )`

Agrega score por llegar a la meta con bonus (coins)

Definition at line 298 of file game_data.c.

4.33.3.6 game_data_are_goals_full() `bool game_data_are_goals_full ( void )`

Avisa si estan todas las metas completas.

Returns

true Si

false No

Definition at line 418 of file game_data.c.

4.33.3.7 game_data_clear_name() `void game_data_clear_name ( void )`

Limpia el nombre del jugador.

Definition at line 361 of file game_data.c.

4.33.3.8 game_data_draw() `void game_data_draw ( void )`

Grafica datos del juego (HUD pricipalmente)

Definition at line 271 of file game_data.c.

4.33.3.9 game_data_get_diff() `int game_data_get_diff ( void )`

Devuelve dificultad.

Returns

int

Definition at line 356 of file game_data.c.

4.33.3.10 game_data_get_frames() `unsigned long game_data_get_frames ( void )`

Devuelve los frames transcurridos del juego.

Returns

unsigned long Frames transcurridos

Definition at line 341 of file game_data.c.

4.33.3.11 game_data_get_game_over_flag() `bool game_data_get_game_over_flag ( void )`

Devuelve flag de game over.

Returns

true Game over

false No game over

Definition at line 410 of file game_data.c.

4.33.3.12 game_data_get_goal_state() `bool game_data_get_goal_state ( unsigned int goal )`

Revisa si un punto de llegada es valido o no (vacio o lleno)

Parameters

goal	0 a MAX_GOALS-1

Returns

true Invalido

false Valido

Definition at line 393 of file game_data.c.

4.33.3.13 game_data_get_lives() `int game_data_get_lives ( void )`

Devuelve vidas.

Returns

int vidas

Definition at line 277 of file game_data.c.

4.33.3.14 game_data_get_name() `char * game_data_get_name ( void )`

Devuelve puntero al nombre del jugador.

Returns

char*

Definition at line 388 of file game_data.c.

4.33.3.15 game_data_get_old_max_score() `unsigned long long game_data_get_old_max_score ( void )`

Devuelve el score maximo sin actualizar al terminar el juego.

Returns

unsigned long long

Definition at line 436 of file game_data.c.

4.33.3.16 game_data_get_run_number() `int game_data_get_run_number ( void )`

Devuelve el numero de run.

Returns

int Numero de run

Definition at line 314 of file game_data.c.

4.33.3.17 game_data_get_run_time_left() int game_data_get_run_time_left (void)

Devuelve el tiempo restante de la run en segundos.

Returns

int Tiempo restante

Definition at line 324 of file game_data.c.

4.33.3.18 game_data_get_score() unsigned long long game_data_get_score (void)

Devuelve score.

Returns

int

Definition at line 287 of file game_data.c.

4.33.3.19 game_data_get_score_max() unsigned long long game_data_get_score_max (void)

Devuelve el score maximo del jugador actual.

Returns

unsigned long long Score maximo

Definition at line 309 of file game_data.c.

4.33.3.20 game_data_get_timer_in_sec() int game_data_get_timer_in_sec (void)

Devuelve el tiempo transcurrido en segundos.

Returns

int Segundos transcurridos

Definition at line 346 of file game_data.c.

4.33.3.21 game_data_init() `void game_data_init (`

```
    void  )
```

Inicializa datos internos del juego.

Definition at line 209 of file game_data.c.

4.33.3.22 game_data_next_run() `void game_data_next_run (`

```
    void  )
```

Indica que se pase a la siguiente run.

Definition at line 319 of file game_data.c.

4.33.3.23 game_data_overwrite_name() `void game_data_overwrite_name (`

```
    char * name )
```

Sobreescribe el nombre del jugador.

Parameters

name	

Definition at line 366 of file game_data.c.

4.33.3.24 game_data_reset_goals() `void game_data_reset_goals (`

```
    void  )
```

Habilita todos los goals.

Definition at line 403 of file game_data.c.

4.33.3.25 game_data_set_diff() `void game_data_set_diff (`

```
    int diff )
```

Setea dificultad.

Parameters

diff	enum DIFFICULTIES

Definition at line 351 of file game_data.c.

4.33.3.26 game_data_set_goal() `void game_data_set_goal ( unsigned int goal )`

Setea un goal como completado.

Parameters

goal	0 a MAX_GOALS-1

Definition at line 398 of file game_data.c.

4.33.3.27 game_data_set_score_max() `void game_data_set_score_max ( unsigned long long score )`

Carga el score maximo del jugador actual.

Parameters

score	

Definition at line 304 of file game_data.c.

4.33.3.28 game_data_subtract_live() `void game_data_subtract_live ( void )`

Resta una vida.

Definition at line 282 of file game_data.c.

4.33.3.29 game_data_update() `void game_data_update ( void )`

Actualiza datos internos del juego.

Definition at line 227 of file game_data.c.

4.34 game_data.h

Go to the documentation of this file.

```
00001
00012 #ifndef _GAME_DATA_H_
00013 #define _GAME_DATA_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include <stdbool.h>
00020
00021 /*****************************************************************************
00022  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00023  *****************************************************************************/
00024
00025 enum DIFFICULTIES
00026 {
00027     DIFFICULTIES_EASY = 1,
00028     DIFFICULTIES_NORMAL,
00029     DIFFICULTIES_HARD
00030 };
00031
00032 /*****************************************************************************
00033  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00034  *****************************************************************************/
00035
00040 void game_data_init(void);
00041
00046 void game_data_update(void);
00047
00052 void game_data_draw(void);
00053
00059 int game_data_get_lives(void);
00060
00065 void game_data_subtract_live(void);
00066
00072 unsigned long long game_data_get_score(void);
00073
00078 void game_data_add_score(void);
00079
00084 void game_data_add_score_bonus(void);
00085
00091 void game_data_set_score_max(unsigned long long score);
00092
00098 unsigned long long game_data_get_score_max(void);
00099
00105 int game_data_get_run_number(void);
00106
00111 void game_data_next_run(void);
00112
00118 int game_data_get_run_time_left(void);
00119
00124 void game_data_add_run_time_goal(void);
00125
00130 void game_data_add_run_time_goal_bonus(void);
00131
00137 unsigned long game_data_get_frames(void);
00138
00144 int game_data_get_timer_in_sec(void);
00145
00151 void game_data_set_diff(int diff);
00152
00158 int game_data_get_diff(void);
00163 void game_data_clear_name(void);
00164
00170 void game_data_overwrite_name(char *name);
00171
00177 void game_data_add_name_letter(char letter);
00178
00184 char *game_data_get_name(void);
00185
00193 bool game_data_get_goal_state(unsigned int goal);
00194
00200 void game_data_set_goal(unsigned int goal);
00201
00206 void game_data_reset_goals(void);
00207
00214 bool game_data_get_game_over_flag(void);
00215
00222 bool game_data_are_goals_full(void);
00223
00229 unsigned long long game_data_get_old_max_score(void);
00230
00231 /*****************************************************************************
```

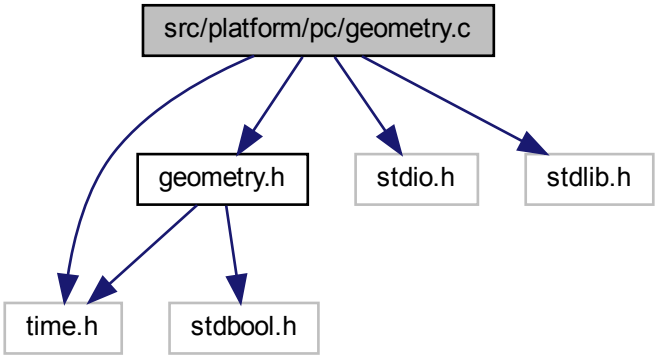
```
00232  ******************************************************************************/
00233
00234 #endif // _GAME_DATA_H_
```

4.35 src/platform/pc/geometry.c File Reference

Source del modulo geometry. Look-up tables con medidas, funciones orientadas a temas geométricos dentro del juego en PC.

```
#include "geometry.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for geometry.c:



Functions

- int get_rand_between (int low, int high)

  Devuelve un randon entre dos numeros dados.
- bool collide (int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)

  Comprueba colición de hitboxes rectangulares.
- bool collideShort (int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)

  Similar a 'collide', pero pide ancho y alto en vez de segundas coordenadas.
- bool inside (int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)

  Detecta si un rectángulo está dentro de otro.
- bool insideShort (int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)

  Similar a 'inside', pero pide ancho y alto en vez de segundas coordenadas.
- bool insideShortScaled (int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh, float scale)

  Similar a 'insideShort', pero permite setear cuan dentro debe estar una box dentro de la otra para tomarla como tal.
- int mapInt (int source, int min_in, int max_in, int min_out, int max_out)

  Toma un valor comprendido dentro de un rango (in) y lo devuelve (mapea) a otro rango (out)
- pair_xy_t getXYFromFrogFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de la rana.
- pair_xy_t getXYFromTurtleFrame (int frame)

Devuelve par de coordenadas xy topleft de un frame dado del sprite de la tortuga.

- pair_xy_t getXYFromCarFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite del auto.
- pair_xy_t getXYFromCoinFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de coin.
- pair_xy_t getXYFromSplashFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de splash.

Variables

- const unsigned int lanes_logs [LANES_LOG_TOTAL] = {2, 4, 5}

  Filas para troncos.
- const unsigned int lanes_turtles [LANES_TURTLE_TOTAL] = {3, 6}

  Filas para tortugas.
- const unsigned int lanes_cars [LANES_CAR_TOTAL] = {8, 9, 10, 11, 12}

  Filas para autos.
- const unsigned int goal_cols [MAX_GOALS] = {1, 4, 7, 10, 13}

  Columnas para puntos de llegada.

4.35.1 Detailed Description

Source del modulo geometry. Look-up tables con medidas, funciones orientadas a temas geométricos dentro del juego en PC.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file geometry.c.

4.35.2 Function Documentation

4.35.2.1 collide()

```
bool collide (
        int ax1,
        int ay1,
        int ax2,
        int ay2,
        int bx1,
        int by1,
        int bx2,
        int by2 )
```

Comprueba colición de hitboxes rectangulares.

Parameters

ax1	topleft corner de a (x)
ay1	topleft corner de a (y)
ax2	bottomright corner de a (x)
ay2	bottomright corner de a (y)
bx1	topleft corner de b (x)
by1	topleft corner de b (y)
bx2	bottomright corner de b (x)
by2	bottomright corner de b (y)

Returns

true Colisión

false No colisión

Definition at line 121 of file geometry.c.

4.35.2.2 collideShort()

```
bool collideShort (
        int ax,
        int ay,
        int aw,
        int ah,
        int bx,
        int by,
        int bw,
        int bh )
```

Similar a 'collide', pero pide ancho y alto en vez de segundas coordenadas.

Parameters

ax	topleft corner x de a
ay	topleft corner y de a
aw	ancho de a
ah	alto de a
bx	topleft corner x de b
by	topleft corner y de b
bw	ancho de b
bh	alto de b

Returns

true Colision

false No colision

Definition at line 126 of file geometry.c.

4.35.2.3 get_rand_between()

```
int get_rand_between (
        int low,
        int high )
```

Devuelve un randon entre dos numeros dados.

Parameters

low	Valor inferior
high	Valor superior

Returns

int Valor random

Definition at line 116 of file geometry.c.

4.35.2.4 getXYFromCarFrame()

```
pair_xy_t getXYFromCarFrame (
        int frame )
```

Devuelve par de coordenadas xy topleft de un frame dado del sprite del auto.

Parameters

frame	Numero de frame (0 a CAR_TYPE_N - 1)

Returns

pair_xy_t Par de coordenandas

Definition at line 172 of file geometry.c.

4.35.2.5 getXYFromCoinFrame()

```
pair_xy_t getXYFromCoinFrame (
        int frame )
```

Devuelve par de coordenadas xy topleft de un frame dado del sprite de coin.

Parameters

frame	Numero de frame (0 a SPRITE_COIN_FRAMES - 1)

Returns

pair_xy_t Par de coordenandas

Definition at line 177 of file geometry.c.

4.35.2.6 getXYFromFrogFrame() pair_xy_t getXYFromFrogFrame (int *frame*)

Devuelve par de coordenadas xy topleft de un frame dado del sprite de la rana.

Parameters

frame	Numero de frame (0 a FROG_FRAMES - 1)

Returns

pair_xy_t Par de coordenadas

Definition at line 162 of file geometry.c.

4.35.2.7 getXYFromSplashFrame() pair_xy_t getXYFromSplashFrame (int *frame*)

Devuelve par de coordenadas xy topleft de un frame dado del sprite de splash.

Parameters

frame	Numero de frame (0 a SPRITE_SPLASH_FRAMES - 1)

Returns

pair_xy_t Par de coordenandas

Definition at line 182 of file geometry.c.

4.35.2.8 getXYFromTurtleFrame() pair_xy_t getXYFromTurtleFrame (int *frame*)

Devuelve par de coordenadas xy topleft de un frame dado del sprite de la tortuga.

Parameters

frame	Numero de frame (0 a TURTLE_FRAMES - 1)

Returns

pair_xy_t Par de coordenadas

Definition at line 167 of file geometry.c.

4.35.2.9 inside()

```
bool inside (
        int ax1,
        int ay1,
        int ax2,
        int ay2,
        int bx1,
        int by1,
        int bx2,
        int by2 )
```

Detecta si un rectángulo está dentro de otro.

Parameters

ax1	topleft corner de big (x)
ay1	topleft corner de big (y)
ax2	bottomright corner de big (x)
ay2	bottomright corner de big (y)
bx1	topleft corner de small (x)
by1	topleft corner de small (y)
bx2	bottomright corner de small (x)
by2	bottomright corner de small (y)

Returns

true Está dentro

false Está fuera

Definition at line 131 of file geometry.c.

4.35.2.10 insideShort()

```
bool insideShort (
        int ax,
        int ay,
        int aw,
        int ah,
        int bx,
        int by,
        int bw,
        int bh )
```

Similar a 'inside', pero pide ancho y alto en vez de segundas coordenadas.

Parameters

ax	topleft corner x de big
ay	topleft corner y de big

Parameters

aw	ancho de big
ah	alto de big
bx	topleft corner x de big
by	topleft corner y de big
bw	ancho de big
bh	alto de big

Returns

true Esta dentro

false Esta fuera

Definition at line 136 of file geometry.c.

4.35.2.11 insideShortScaled()

```
bool insideShortScaled (
    int ax,
    int ay,
    int aw,
    int ah,
    int bx,
    int by,
    int bw,
    int bh,
    float scale )
```

Similar a 'insideShort', pero permite setear cuan dentro debe estar una box dentro de la otra para tomarla como tal.

Parameters

ax	topleft corner x de big
ay	topleft corner y de big
aw	ancho de big
ah	alto de big
bx	topleft corner x de big
by	topleft corner y de big
bw	ancho de big
bh	alto de big
scale	Factor de insercion. Entre 0.0 (nada metido) y 1.0 (completamente metido). Otro valor devuelve false

Returns

true

false

Definition at line 141 of file geometry.c.

4.35.2.12 mapInt() `int mapInt (`

```
int source,
int min_in,
int max_in,
int min_out,
int max_out )
```

Toma un valor comprendido dentro de un rango (in) y lo devuelve (mapea) a otro rango (out)

@source https://stackoverflow.com/questions/5731863/mapping-a-numeric-range-onto-another

Parameters

source	Valor a mapear
min_in	Limite inferior del rango de entrada
max_in	Límite superior del rango de entrada
min_out	Límite inferior del rango de salida
max_out	Límite superior del rango de salida

Returns

int Valor mapeado

Definition at line 151 of file geometry.c.

4.35.3 Variable Documentation

4.35.3.1 goal_cols `const unsigned int goal_cols[MAX_GOALS] = {1, 4, 7, 10, 13}`

Columnas para puntos de llegada.

Columnas para puntos de llegada, referenciadas a 0.

Definition at line 47 of file geometry.c.

4.35.3.2 lanes_cars `const unsigned int lanes_cars[LANES_CAR_TOTAL] = {8, 9, 10, 11, 12}`

Filas para autos.

Filas para autos, referenciadas a 0.

Definition at line 41 of file geometry.c.

4.35.3.3 lanes_logs `const unsigned int lanes_logs[LANES_LOG_TOTAL] = {2, 4, 5}`

Filas para troncos.

Filas para troncos, referenciadas a 0.

Definition at line 29 of file geometry.c.

4.35.3.4 lanes_turtles `const unsigned int lanes_turtles[LANES_TURTLE_TOTAL] = {3, 6}`

Filas para tortugas.

Filas para tortugas, referenciadas a 0.

Definition at line 35 of file geometry.c.

4.36 geometry.c

Go to the documentation of this file.

```
00001
00012 /**********************************************************************************
00013  * INCLUDE HEADER FILES
00014  **********************************************************************************/
00015
00016 #include "geometry.h"
00017 #include <stdio.h>
00018 #include <stdlib.h>
00019 #include <time.h>
00020
00021 /**********************************************************************************
00022  * VARIABLES WITH GLOBAL SCOPE
00023  **********************************************************************************/
00024
00029 const unsigned int lanes_logs[LANES_LOG_TOTAL] = {2, 4, 5};
00030
00035 const unsigned int lanes_turtles[LANES_TURTLE_TOTAL] = {3, 6};
00036
00041 const unsigned int lanes_cars[LANES_CAR_TOTAL] = {8, 9, 10, 11, 12};
00042
00047 const unsigned int goal_cols[MAX_GOALS] = {1, 4, 7, 10, 13};
00048
00049 /**********************************************************************************
00050  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00051  **********************************************************************************/
00052
00053 // coordenadas topleft de cada frame del sprite de la rana
00054 static const pair_xy_t pair_xy_frog_sprites_frames[FROG_FRAMES] =
00055     {
00056         {16, 16},
00057         {79, 15},
00058         {16, 79},
00059         {65, 79},
00060         {14, 141},
00061         {76, 141},
00062         {16, 204},
00063         {79, 190}};
00064
00065 // coordenadas topleft de cada frame del sprite de la tortuga
00066 static const pair_xy_t pair_xy_turtle_sprites_frames[TURTLE_FRAMES] =
00067     {
00068         {2, 0},
00069         {51, 0},
00070         {99, 0},
00071         {146, 0},
00072         {194, 0},
00073         {243, 0},
00074         {290, 0},
00075         {337, 0},
00076         {382, 0},
00077         {445, 0},
00078         {496, 0}};
```

```
00079
00080 // coordenadas topleft de cada frame del sprite del auto
00081 static const pair_xy_t pair_xy_car_sprites_frames[CAR_TYPE_N] =
00082     {
00083         {0, 1},   // azul
00084         {77, 1},  // policia
00085         {155, 1}, // amarillo
00086         {0, 56},  // fire truck
00087         {0, 112}  // truck
00088 };
00089
00090 // idem para sprite de coin
00091 static const pair_xy_t pair_xy_coin_sprites_frames[SPRITE_COIN_FRAMES] =
00092     {
00093         {0, 0},
00094         {25, 0},
00095         {51, 0},
00096         {76, 0},
00097         {104, 0},
00098         {131, 0}};
00099
00100 // idem para sprite de splash
00101 static const pair_xy_t pair_xy_splash_sprites_frames[SPRITE_SPLASH_FRAMES] =
00102     {
00103         {0, 0},
00104         {105, 0},
00105         {206, 0},
00106         {0, 86},
00107         {103, 86},
00108         {206, 86}};
00109
00110 /*****************************************************************************
00111  *****************************************************************************
00112             GLOBAL FUNCTION DEFINITIONS
00113  *****************************************************************************
00114  *****************************************************************************/
00115
00116 int get_rand_between(int low, int high)
00117 {
00118     return (rand() % ((high + 1) - low) + low);
00119 }
00120
00121 bool collide(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)
00122 {
00123     return ax1 < bx2 && ax2 > bx1 && ay1 < by2 && ay2 > by1;
00124 }
00125
00126 bool collideShort(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)
00127 {
00128     return (collide(ax, ay, ax + aw, ay + ah, bx, by, bx + bw, by + bh));
00129 }
00130
00131 bool inside(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)
00132 {
00133     return bx1 > ax1 && by1 > ay1 && bx2 < ax2 && by2 < ay2;
00134 }
00135
00136 bool insideShort(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)
00137 {
00138     return (inside(ax, ay, ax + aw, ay + ah, bx, by, bx + bw, by + bh));
00139 }
00140
00141 bool insideShortScaled(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh, float scale)
00142 {
00143     if (scale < 0.0 || scale > 1.0)
00144         return false;
00145
00146     float diff = (bw * (1.0 - scale));
00147
00148     return (inside(ax, ay, ax + aw, ay + ah, bx + diff, by, bx + bw - diff, by + bh));
00149 }
00150
00151 int mapInt(int source, int min_in, int max_in, int min_out, int max_out)
00152 {
00153     // int slope = (max_out - max_in) / (min_out - min_in);
00154
00155     // int output = min_out + (slope * (source - min_in));
00156
00157     int output = (source - min_in) * (max_out - min_out) / (max_in - min_in) + min_out;
00158
00159     return (output);
00160 }
00161
00162 pair_xy_t getXYFromFrogFrame(int frame)
00163 {
00164     return (pair_xy_frog_sprites_frames[frame]);
00165 }
```

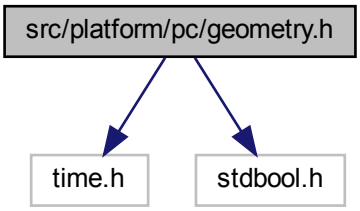
```
00166
00167 pair_xy_t getXYFromTurtleFrame(int frame)
00168 {
00169     return (pair_xy_turtle_sprites_frames[frame]);
00170 }
00171
00172 pair_xy_t getXYFromCarFrame(int frame)
00173 {
00174     return (pair_xy_car_sprites_frames[frame]);
00175 }
00176
00177 pair_xy_t getXYFromCoinFrame(int frame)
00178 {
00179     return (pair_xy_coin_sprites_frames[frame]);
00180 }
00181
00182 pair_xy_t getXYFromSplashFrame(int frame)
00183 {
00184     return (pair_xy_splash_sprites_frames[frame]);
00185 }
```

4.37 src/platform/pc/geometry.h File Reference

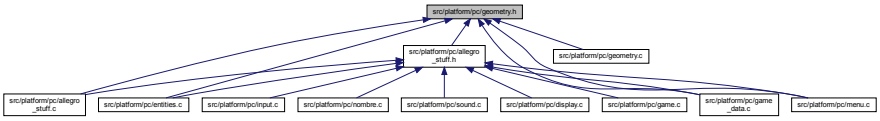
Header del modulo geometry. Defines y enums relacionados a medidas, cantidades y estados para la implementación en PC.

```
#include <time.h>
#include <stdbool.h>
```

Include dependency graph for geometry.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct pair_xy_t

Macros

- #define DISPLAY_W 690
- #define DISPLAY_H 644
- #define ROWS 14
- #define COLS 15
- #define CELL_H 46
- #define CELL_W 46
- #define MAX_LANES (ROWS - 1)
- #define LANES_CAR_TOTAL 5
- #define LANES_LOG_TOTAL 3
- #define LANES_TURTLE_TOTAL 2
- #define CELL_TOPLEFT_X 0
- #define CELL_TOPLEFT_Y 0
- #define CELL_START_X (CELL_TOPLEFT_X + CELL_W ∗ ((COLS - 1) / 2))
- #define CELL_START_Y (CELL_TOPLEFT_Y + CELL_H ∗ (ROWS - 1))
- #define FROG_W 30
- #define FROG_H 30
- #define FROG_OFFSET_X (CELL_W / 2 - FROG_W / 2)
- #define FROG_OFFSET_Y (CELL_H / 2 - FROG_H / 2)
- #define CELL_START_FROG_X (CELL_START_X + FROG_OFFSET_X)
- #define CELL_START_FROG_Y (CELL_START_Y + FROG_OFFSET_Y)
- #define FROG_FRAMES 8
- #define SPRITE_SIZE_FROG_STATIC_H FROG_H
- #define SPRITE_SIZE_FROG_STATIC_W FROG_W
- #define SPRITE_SIZE_FROG_DYNAMIC_LONG 46
- #define SPRITE_SIZE_FROG_DYNAMIC_SHORT FROG_W
- #define STEP_FULL_SIZE CELL_H
- #define STEP_RATIO (CELL_H / 3)
- #define STEP_FRACTION_SIZE (STEP_FULL_SIZE / STEP_RATIO)
- #define FROG_MAX_X (DISPLAY_W - (CELL_W - FROG_OFFSET_X))
- #define FROG_MAX_Y (DISPLAY_H - (CELL_H - FROG_OFFSET_Y))
- #define FROG_MIN_X (CELL_TOPLEFT_X + FROG_OFFSET_X)
- #define FROG_MIN_Y (CELL_TOPLEFT_Y + FROG_OFFSET_Y + CELL_H)
- #define LOG_W (4 ∗ CELL_W)
- #define LOG_H 40
- #define LOG_OFFSET_X 0
- #define LOG_OFFSET_Y (CELL_H / 2 - LOG_H / 2)
- #define CAR_W CELL_W + 26
- #define CAR_TRUCK_FIRE_W (3 ∗ CELL_W)
- #define CAR_TRUCK_W (4 ∗ CELL_W)
- #define CAR_H 40
- #define CAR_OFFSET_X 0
- #define CAR_OFFSET_Y (CELL_H / 2 - CAR_H / 2)
- #define TURTLE_FRAMES 11
- #define TURTLE_SIDE CELL_W
- #define TURTLE_FRAME_OFFSET_XY (CELL_W / 2 - TURTLE_SIDE / 2)
- #define SPRITE_SIZE_HEART 25
- #define SPRITE_DEAD_SIZE 35
- #define SPRITE_DEAD_OFFSET (CELL_W / 2 - SPRITE_DEAD_SIZE / 2)
- #define SPRITE_COIN_FRAMES 6
- #define SPRITE_COIN_SIDE 24
- #define SPRITE_COIN_OFFSET_XY (CELL_W / 2 - SPRITE_COIN_SIDE / 2)
- #define SPRITE_SPLASH_FRAMES 6
- #define SPRITE_SPLASH_W 98

- #define SPRITE_SPLASH_H 68
- #define SPRITE_SPLASH_OFFSET_X (CELL_W / 2 - SPRITE_SPLASH_W / 2)
- #define SPRITE_SPLASH_OFFSET_Y (CELL_W / 2 - SPRITE_SPLASH_H / 2)
- #define SPRITE_BORDER_START_X 0
- #define SPRITE_BORDER_START_Y CELL_H
- #define MENU_OPTION_TOPLEFT_X 45
- #define MENU_OPTION_TOPLEFT_Y 72
- #define MENU_OPTION_DELTA_Y 100
- #define MENU_OPTION_W 600
- #define MENU_OPTION_H 75
- #define CREDITS_SCREEN_LENGTH 2576
- #define CREDITS_SCREEN_START 0
- #define CREDITS_SCREEN_FINAL (CREDITS_SCREEN_LENGTH - DISPLAY_H)
- #define INSERTION_FACTOR (double)0.5
- #define GOAL_ROW_OFFSET_Y_FIX 5
- #define GOAL_ROW_MARGIN_TO_REACH 5

Typedefs

- typedef enum CAR_TYPE **CAR_TYPE**

Enumerations

- enum **GOALS** {
  GOAL_LEFT , **GOAL_LEFT_MID** , **GOAL_MID** , **GOAL_RIGHT_MID** ,
  GOAL_RIGHT , **MAX_GOALS** }
- enum **DIRECTIONS** {
  DIRECTION_NONE , **DIRECTION_UP** , **DIRECTION_RIGHT** , **DIRECTION_LEFT** ,
  DIRECTION_DOWN }
- enum **MENU_STATES** {
  MENU_STATE_OPCION_0 , **MENU_STATE_OPCION_1** , **MENU_STATE_OPCION_2** , **MENU_STATE_↩**
  OPCION_3 ,
  MENU_STATE_OPCION_4 , **MENU_STATE_MAX** }
- enum **MENU_WINDOWS** {
  MENU_WINDOW_HOME , **MENU_WINDOW_DIFFICULTY** , **MENU_WINDOW_PAUSE** , **MENU_↩**
  WINDOW_GAME_OVER ,
  MENU_WINDOW_MAX }
- enum **CAR_TYPE** {
  CAR_BLUE = 0 , **CAR_POLICE** , **CAR_YELLOW** , **TRUCK_FIRE** ,
  TRUCK , **CAR_TYPE_N** }

Functions

- int get_rand_between (int low, int high)

  Devuelve un randon entre dos numeros dados.
- bool collide (int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)

  Comprueba colición de hitboxes rectangulares.
- bool collideShort (int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)

  Similar a 'collide', pero pide ancho y alto en vez de segundas coordenadas.
- bool inside (int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)

  Detecta si un rectángulo está dentro de otro.
- bool insideShort (int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)

Similar a 'inside', pero pide ancho y alto en vez de segundas coordenadas.

- bool insideShortScaled (int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh, float scale)

  Similar a 'insideShort', pero permite setear cuan dentro debe estar una box dentro de la otra para tomarla como tal.
- int mapInt (int source, int min_in, int max_in, int min_out, int max_out)

  Toma un valor comprendido dentro de un rango (in) y lo devuelve (mapea) a otro rango (out)
- pair_xy_t getXYFromFrogFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de la rana.
- pair_xy_t getXYFromTurtleFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de la tortuga.
- pair_xy_t getXYFromCarFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite del auto.
- pair_xy_t getXYFromCoinFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de coin.
- pair_xy_t getXYFromSplashFrame (int frame)

  Devuelve par de coordenadas xy topleft de un frame dado del sprite de splash.

Variables

- const unsigned int lanes_logs [LANES_LOG_TOTAL]

  Filas para troncos, referenciadas a 0.
- const unsigned int lanes_turtles [LANES_TURTLE_TOTAL]

  Filas para tortugas, referenciadas a 0.
- const unsigned int lanes_cars [LANES_CAR_TOTAL]

  Filas para autos, referenciadas a 0.
- const unsigned int goal_cols [MAX_GOALS]

  Columnas para puntos de llegada, referenciadas a 0.

4.37.1 Detailed Description

Header del modulo geometry. Defines y enums relacionados a medidas, cantidades y estados para la implementación en PC.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file geometry.h.

4.37.2 Macro Definition Documentation

4.37.2.1 CAR_H `#define CAR_H 40`

Definition at line 87 of file geometry.h.

4.37.2.2 CAR_OFFSET_X `#define CAR_OFFSET_X 0`

Definition at line 88 of file geometry.h.

4.37.2.3 CAR_OFFSET_Y `#define CAR_OFFSET_Y (CELL_H / 2 - CAR_H / 2)`

Definition at line 89 of file geometry.h.

4.37.2.4 CAR_TRUCK_FIRE_W `#define CAR_TRUCK_FIRE_W (3 * CELL_W)`

Definition at line 85 of file geometry.h.

4.37.2.5 CAR_TRUCK_W `#define CAR_TRUCK_W (4 * CELL_W)`

Definition at line 86 of file geometry.h.

4.37.2.6 CAR_W `#define CAR_W CELL_W + 26`

Definition at line 84 of file geometry.h.

4.37.2.7 CELL_H `#define CELL_H 46`

Definition at line 30 of file geometry.h.

4.37.2.8 CELL_START_FROG_X `#define CELL_START_FROG_X (CELL_START_X + FROG_OFFSET_X)`

Definition at line 56 of file geometry.h.

4.37.2.9 CELL_START_FROG_Y `#define CELL_START_FROG_Y (CELL_START_Y + FROG_OFFSET_Y)`

Definition at line 57 of file geometry.h.

4.37.2.10 CELL_START_X `#define CELL_START_X (CELL_TOPLEFT_X + CELL_W * ((COLS - 1) / 2))`

Definition at line 46 of file geometry.h.

4.37.2.11 CELL_START_Y `#define CELL_START_Y (CELL_TOPLEFT_Y + CELL_H * (ROWS - 1))`

Definition at line 47 of file geometry.h.

4.37.2.12 CELL_TOPLEFT_X `#define CELL_TOPLEFT_X 0`

Definition at line 42 of file geometry.h.

4.37.2.13 CELL_TOPLEFT_Y `#define CELL_TOPLEFT_Y 0`

Definition at line 43 of file geometry.h.

4.37.2.14 CELL_W `#define CELL_W 46`

Definition at line 31 of file geometry.h.

4.37.2.15 COLS `#define COLS 15`

Definition at line 29 of file geometry.h.

4.37.2.16 CREDITS_SCREEN_FINAL `#define CREDITS_SCREEN_FINAL (CREDITS_SCREEN_LENGTH - DISPLAY↩ _H)`

Definition at line 123 of file geometry.h.

4.37.2.17 CREDITS_SCREEN_LENGTH `#define CREDITS_SCREEN_LENGTH 2576`

Definition at line 121 of file geometry.h.

4.37.2.18 CREDITS_SCREEN_START `#define CREDITS_SCREEN_START 0`

Definition at line 122 of file geometry.h.

4.37.2.19 DISPLAY_H `#define DISPLAY_H 644`

Definition at line 27 of file geometry.h.

4.37.2.20 DISPLAY_W `#define DISPLAY_W 690`

Definition at line 26 of file geometry.h.

4.37.2.21 FROG_FRAMES `#define FROG_FRAMES 8`

Definition at line 60 of file geometry.h.

4.37.2.22 FROG_H `#define FROG_H 30`

Definition at line 50 of file geometry.h.

4.37.2.23 FROG_MAX_X `#define FROG_MAX_X (DISPLAY_W - (CELL_W - FROG_OFFSET_X))`

Definition at line 72 of file geometry.h.

4.37.2.24 FROG_MAX_Y `#define FROG_MAX_Y (DISPLAY_H - (CELL_H - FROG_OFFSET_Y))`

Definition at line 73 of file geometry.h.

4.37.2.25 FROG_MIN_X `#define FROG_MIN_X (CELL_TOPLEFT_X + FROG_OFFSET_X)`

Definition at line 74 of file geometry.h.

4.37.2.26 FROG_MIN_Y `#define FROG_MIN_Y (CELL_TOPLEFT_Y + FROG_OFFSET_Y + CELL_H)`

Definition at line 75 of file geometry.h.

4.37.2.27 FROG_OFFSET_X `#define FROG_OFFSET_X (CELL_W / 2 - FROG_W / 2)`

Definition at line 52 of file geometry.h.

4.37.2.28 FROG_OFFSET_Y `#define FROG_OFFSET_Y (CELL_H / 2 - FROG_H / 2)`

Definition at line 53 of file geometry.h.

4.37.2.29 FROG_W `#define FROG_W 30`

Definition at line 49 of file geometry.h.

4.37.2.30 GOAL_ROW_MARGIN_TO_REACH `#define GOAL_ROW_MARGIN_TO_REACH 5`

Definition at line 129 of file geometry.h.

4.37.2.31 GOAL_ROW_OFFSET_Y_FIX `#define GOAL_ROW_OFFSET_Y_FIX 5`

Definition at line 128 of file geometry.h.

4.37.2.32 INSERTION_FACTOR `#define INSERTION_FACTOR (double)0.5`

Definition at line 126 of file geometry.h.

4.37.2.33 LANES_CAR_TOTAL `#define LANES_CAR_TOTAL 5`

Definition at line 35 of file geometry.h.

4.37.2.34 LANES_LOG_TOTAL `#define LANES_LOG_TOTAL 3`

Definition at line 37 of file geometry.h.

4.37.2.35 LANES_TURTLE_TOTAL `#define LANES_TURTLE_TOTAL 2`

Definition at line 39 of file geometry.h.

4.37.2.36 LOG_H `#define LOG_H 40`

Definition at line 79 of file geometry.h.

4.37.2.37 LOG_OFFSET_X `#define LOG_OFFSET_X 0`

Definition at line 80 of file geometry.h.

4.37.2.38 LOG_OFFSET_Y `#define LOG_OFFSET_Y (CELL_H / 2 - LOG_H / 2)`

Definition at line 81 of file geometry.h.

4.37.2.39 LOG_W `#define LOG_W (4 * CELL_W)`

Definition at line 78 of file geometry.h.

4.37.2.40 MAX_LANES `#define MAX_LANES (ROWS - 1)`

Definition at line 33 of file geometry.h.

4.37.2.41 MENU_OPTION_DELTA_Y `#define MENU_OPTION_DELTA_Y 100`

Definition at line 117 of file geometry.h.

4.37.2.42 MENU_OPTION_H `#define MENU_OPTION_H 75`

Definition at line 119 of file geometry.h.

4.37.2.43 MENU_OPTION_TOPLEFT_X `#define MENU_OPTION_TOPLEFT_X 45`

Definition at line 115 of file geometry.h.

4.37.2.44 MENU_OPTION_TOPLEFT_Y `#define MENU_OPTION_TOPLEFT_Y 72`

Definition at line 116 of file geometry.h.

4.37.2.45 MENU_OPTION_W `#define MENU_OPTION_W 600`

Definition at line 118 of file geometry.h.

4.37.2.46 ROWS `#define ROWS 14`

Definition at line 28 of file geometry.h.

4.37.2.47 SPRITE_BORDER_START_X `#define SPRITE_BORDER_START_X 0`

Definition at line 112 of file geometry.h.

4.37.2.48 SPRITE_BORDER_START_Y `#define SPRITE_BORDER_START_Y CELL_H`

Definition at line 113 of file geometry.h.

4.37.2.49 SPRITE_COIN_FRAMES `#define SPRITE_COIN_FRAMES 6`

Definition at line 102 of file geometry.h.

4.37.2.50 SPRITE_COIN_OFFSET_XY `#define SPRITE_COIN_OFFSET_XY (CELL_W / 2 - SPRITE_COIN_↩ SIDE / 2)`

Definition at line 104 of file geometry.h.

4.37.2.51 SPRITE_COIN_SIDE `#define SPRITE_COIN_SIDE 24`

Definition at line 103 of file geometry.h.

4.37.2.52 SPRITE_DEAD_OFFSET `#define SPRITE_DEAD_OFFSET (CELL_W / 2 - SPRITE_DEAD_SIZE / 2)`

Definition at line 100 of file geometry.h.

4.37.2.53 SPRITE_DEAD_SIZE `#define SPRITE_DEAD_SIZE 35`

Definition at line 99 of file geometry.h.

4.37.2.54 SPRITE_SIZE_FROG_DYNAMIC_LONG `#define SPRITE_SIZE_FROG_DYNAMIC_LONG 46`

Definition at line 63 of file geometry.h.

4.37.2.55 SPRITE_SIZE_FROG_DYNAMIC_SHORT `#define SPRITE_SIZE_FROG_DYNAMIC_SHORT FROG_W`

Definition at line 64 of file geometry.h.

4.37.2.56 SPRITE_SIZE_FROG_STATIC_H `#define SPRITE_SIZE_FROG_STATIC_H FROG_H`

Definition at line 61 of file geometry.h.

4.37.2.57 SPRITE_SIZE_FROG_STATIC_W `#define SPRITE_SIZE_FROG_STATIC_W FROG_W`

Definition at line 62 of file geometry.h.

4.37.2.58 SPRITE_SIZE_HEART `#define SPRITE_SIZE_HEART 25`

Definition at line 97 of file geometry.h.

4.37.2.59 SPRITE_SPLASH_FRAMES `#define SPRITE_SPLASH_FRAMES 6`

Definition at line 106 of file geometry.h.

4.37.2.60 SPRITE_SPLASH_H `#define SPRITE_SPLASH_H 68`

Definition at line 108 of file geometry.h.

4.37.2.61 SPRITE_SPLASH_OFFSET_X `#define SPRITE_SPLASH_OFFSET_X (CELL_W / 2 - SPRITE_↩ SPLASH_W / 2)`

Definition at line 109 of file geometry.h.

4.37.2.62 SPRITE_SPLASH_OFFSET_Y `#define SPRITE_SPLASH_OFFSET_Y (CELL_W / 2 - SPRITE_↩ SPLASH_H / 2)`

Definition at line 110 of file geometry.h.

4.37.2.63 SPRITE_SPLASH_W `#define SPRITE_SPLASH_W 98`

Definition at line 107 of file geometry.h.

4.37.2.64 STEP_FRACTION_SIZE `#define STEP_FRACTION_SIZE (STEP_FULL_SIZE / STEP_RATIO)`

Definition at line 69 of file geometry.h.

4.37.2.65 STEP_FULL_SIZE `#define STEP_FULL_SIZE CELL_H`

Definition at line 67 of file geometry.h.

4.37.2.66 STEP_RATIO `#define STEP_RATIO (CELL_H / 3)`

Definition at line 68 of file geometry.h.

4.37.2.67 TURTLE_FRAME_OFFSET_XY `#define TURTLE_FRAME_OFFSET_XY (CELL_W / 2 - TURTLE_SIDE / 2)`

Definition at line 94 of file geometry.h.

4.37.2.68 TURTLE_FRAMES `#define TURTLE_FRAMES 11`

Definition at line 92 of file geometry.h.

4.37.2.69 TURTLE_SIDE `#define TURTLE_SIDE CELL_W`

Definition at line 93 of file geometry.h.

4.37.3 Enumeration Type Documentation

4.37.3.1 CAR_TYPE `enum CAR_TYPE`

Definition at line 180 of file geometry.h.

4.37.3.2 DIRECTIONS `enum DIRECTIONS`

Definition at line 152 of file geometry.h.

4.37.3.3 GOALS `enum GOALS`

Definition at line 142 of file geometry.h.

4.37.3.4 MENU_STATES enum MENU_STATES

Definition at line 161 of file geometry.h.

4.37.3.5 MENU_WINDOWS enum MENU_WINDOWS

Definition at line 171 of file geometry.h.

4.37.4 Function Documentation

4.37.4.1 collide() bool collide (

```
int ax1,
int ay1,
int ax2,
int ay2,
int bx1,
int by1,
int bx2,
int by2 )
```

Comprueba colición de hitboxes rectangulares.

Parameters

ax1	topleft corner de a (x)
ay1	topleft corner de a (y)
ax2	bottomright corner de a (x)
ay2	bottomright corner de a (y)
bx1	topleft corner de b (x)
by1	topleft corner de b (y)
bx2	bottomright corner de b (x)
by2	bottomright corner de b (y)

Returns

true Colisión

false No colisión

Definition at line 121 of file geometry.c.

4.37.4.2 collideShort() `bool collideShort (`

```
int ax,
int ay,
int aw,
int ah,
int bx,
int by,
int bw,
int bh )
```

Similar a 'collide', pero pide ancho y alto en vez de segundas coordenadas.

Parameters

ax	topleft corner x de a
ay	topleft corner y de a
aw	ancho de a
ah	alto de a
bx	topleft corner x de b
by	topleft corner y de b
bw	ancho de b
bh	alto de b

Returns

true Colision

false No colision

Definition at line 126 of file geometry.c.

4.37.4.3 get_rand_between() `int get_rand_between (`

```
int low,
int high )
```

Devuelve un randon entre dos numeros dados.

Parameters

low	Valor inferior
high	Valor superior

Returns

int Valor random

Definition at line 116 of file geometry.c.

4.37.4.4 getXYFromCarFrame() pair_xy_t getXYFromCarFrame (int frame)

Devuelve par de coordenadas xy topleft de un frame dado del sprite del auto.

Parameters

frame	Numero de frame (0 a CAR_TYPE_N - 1)

Returns

pair_xy_t Par de coordenandas

Definition at line 172 of file geometry.c.

4.37.4.5 getXYFromCoinFrame() pair_xy_t getXYFromCoinFrame (int frame)

Devuelve par de coordenadas xy topleft de un frame dado del sprite de coin.

Parameters

frame	Numero de frame (0 a SPRITE_COIN_FRAMES - 1)

Returns

pair_xy_t Par de coordenandas

Definition at line 177 of file geometry.c.

4.37.4.6 getXYFromFrogFrame() pair_xy_t getXYFromFrogFrame (int frame)

Devuelve par de coordenadas xy topleft de un frame dado del sprite de la rana.

Parameters

frame	Numero de frame (0 a FROG_FRAMES - 1)

Returns

pair_xy_t Par de coordenadas

Definition at line 162 of file geometry.c.

4.37.4.7 getXYFromSplashFrame() `pair_xy_t getXYFromSplashFrame (`

```
int frame )
```

Devuelve par de coordenadas xy topleft de un frame dado del sprite de splash.

Parameters

frame	Numero de frame (0 a SPRITE_SPLASH_FRAMES - 1)

Returns

pair_xy_t Par de coordenandas

Definition at line 182 of file geometry.c.

4.37.4.8 getXYFromTurtleFrame() `pair_xy_t getXYFromTurtleFrame (`

```
int frame )
```

Devuelve par de coordenadas xy topleft de un frame dado del sprite de la tortuga.

Parameters

frame	Numero de frame (0 a TURTLE_FRAMES - 1)

Returns

pair_xy_t Par de coordenadas

Definition at line 167 of file geometry.c.

4.37.4.9 inside() `bool inside (`

```
int ax1,
int ay1,
int ax2,
int ay2,
int bx1,
int by1,
int bx2,
int by2 )
```

Detecta si un rectángulo está dentro de otro.

Parameters

ax1	topleft corner de big (x)
ay1	topleft corner de big (y)
ax2	bottomright corner de big (x)

Parameters

ay2	bottomright corner de big (y)
bx1	topleft corner de small (x)
by1	topleft corner de small (y)
bx2	bottomright corner de small (x)
by2	bottomright corner de small (y)

Returns

true Está dentro

false Está fuera

Definition at line 131 of file geometry.c.

4.37.4.10 insideShort()

```
bool insideShort (
        int ax,
        int ay,
        int aw,
        int ah,
        int bx,
        int by,
        int bw,
        int bh )
```

Similar a 'inside', pero pide ancho y alto en vez de segundas coordenadas.

Parameters

ax	topleft corner x de big
ay	topleft corner y de big
aw	ancho de big
ah	alto de big
bx	topleft corner x de big
by	topleft corner y de big
bw	ancho de big
bh	alto de big

Returns

true Esta dentro

false Esta fuera

Definition at line 136 of file geometry.c.

4.37.4.11 insideShortScaled()

```
bool insideShortScaled (
    int ax,
    int ay,
    int aw,
    int ah,
    int bx,
    int by,
    int bw,
    int bh,
    float scale )
```

Similar a 'insideShort', pero permite setear cuan dentro debe estar una box dentro de la otra para tomarla como tal.

Parameters

ax	topleft corner x de big
ay	topleft corner y de big
aw	ancho de big
ah	alto de big
bx	topleft corner x de big
by	topleft corner y de big
bw	ancho de big
bh	alto de big
scale	Factor de insercion. Entre 0.0 (nada metido) y 1.0 (completamente metido). Otro valor devuelve false

Returns

true

false

Definition at line 141 of file geometry.c.

4.37.4.12 mapInt()

```
int mapInt (
    int source,
    int min_in,
    int max_in,
    int min_out,
    int max_out )
```

Toma un valor comprendido dentro de un rango (in) y lo devuelve (mapea) a otro rango (out)

@source https://stackoverflow.com/questions/5731863/mapping-a-numeric-range-onto-another

Parameters

source	Valor a mapear
min_in	Limite inferior del rango de entrada
max_in	Límite superior del rango de entrada
min_out	Límite inferior del rango de salida
max_out	Límite superior del rango de salida

Returns

int Valor mapeado

Definition at line 151 of file geometry.c.

4.37.5 Variable Documentation

4.37.5.1 goal_cols `const unsigned int goal_cols[MAX_GOALS] [extern]`

Columnas para puntos de llegada, referenciadas a 0.

Columnas para puntos de llegada, referenciadas a 0.

Definition at line 47 of file geometry.c.

4.37.5.2 lanes_cars `const unsigned int lanes_cars[LANES_CAR_TOTAL] [extern]`

Filas para autos, referenciadas a 0.

Filas para autos, referenciadas a 0.

Definition at line 41 of file geometry.c.

4.37.5.3 lanes_logs `const unsigned int lanes_logs[LANES_LOG_TOTAL] [extern]`

Filas para troncos, referenciadas a 0.

Filas para troncos, referenciadas a 0.

Definition at line 29 of file geometry.c.

4.37.5.4 lanes_turtles `const unsigned int lanes_turtles[LANES_TURTLE_TOTAL] [extern]`

Filas para tortugas, referenciadas a 0.

Filas para tortugas, referenciadas a 0.

Definition at line 35 of file geometry.c.

4.38 geometry.h

Go to the documentation of this file.

```
00001
00012 #ifndef _GEOMETRY_H_
00013 #define _GEOMETRY_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include <time.h>
00020 #include <stdbool.h>
00021
00022 /*****************************************************************************
00023  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00024  *****************************************************************************/
00025
00026 #define DISPLAY_W 690
00027 #define DISPLAY_H 644
00028 #define ROWS 14
00029 #define COLS 15
00030 #define CELL_H 46
00031 #define CELL_W 46
00032
00033 #define MAX_LANES (ROWS - 1) //-1 por la primera que es para HUD
00034
00035 #define LANES_CAR_TOTAL 5
00036
00037 #define LANES_LOG_TOTAL 3
00038
00039 #define LANES_TURTLE_TOTAL 2
00040
00041 // Coordenadas de la celda topleft (en su vértice topleft)
00042 #define CELL_TOPLEFT_X 0
00043 #define CELL_TOPLEFT_Y 0
00044
00045 // Coordenadas de inicio ("ultima fila, columna 8"; referidas a topleft)
00046 #define CELL_START_X (CELL_TOPLEFT_X + CELL_W * ((COLS - 1) / 2))
00047 #define CELL_START_Y (CELL_TOPLEFT_Y + CELL_H * (ROWS - 1))
00048
00049 #define FROG_W 30
00050 #define FROG_H 30
00051
00052 #define FROG_OFFSET_X (CELL_W / 2 - FROG_W / 2)
00053 #define FROG_OFFSET_Y (CELL_H / 2 - FROG_H / 2)
00054
00055 // Coordenadas inicio rana
00056 #define CELL_START_FROG_X (CELL_START_X + FROG_OFFSET_X)
00057 #define CELL_START_FROG_Y (CELL_START_Y + FROG_OFFSET_Y)
00058
00059 // Para los sprites de la rana
00060 #define FROG_FRAMES 8
00061 #define SPRITE_SIZE_FROG_STATIC_H FROG_H
00062 #define SPRITE_SIZE_FROG_STATIC_W FROG_W
00063 #define SPRITE_SIZE_FROG_DYNAMIC_LONG 46
00064 #define SPRITE_SIZE_FROG_DYNAMIC_SHORT FROG_W
00065
00066 // Para los pasos de la rana
00067 #define STEP_FULL_SIZE CELL_H
00068 #define STEP_RATIO (CELL_H / 3)
00069 #define STEP_FRACTION_SIZE (STEP_FULL_SIZE / STEP_RATIO)
00070
00071 // Bordes para la rana en el mapa
00072 #define FROG_MAX_X (DISPLAY_W - (CELL_W - FROG_OFFSET_X))
00073 #define FROG_MAX_Y (DISPLAY_H - (CELL_H - FROG_OFFSET_Y))
00074 #define FROG_MIN_X (CELL_TOPLEFT_X + FROG_OFFSET_X)
00075 #define FROG_MIN_Y (CELL_TOPLEFT_Y + FROG_OFFSET_Y + CELL_H)
00076
00077 // Troncos
00078 #define LOG_W (4 * CELL_W)
00079 #define LOG_H 40
00080 #define LOG_OFFSET_X 0
00081 #define LOG_OFFSET_Y (CELL_H / 2 - LOG_H / 2)
00082
00083 // Autos
00084 #define CAR_W CELL_W + 26
00085 #define CAR_TRUCK_FIRE_W (3 * CELL_W)
00086 #define CAR_TRUCK_W (4 * CELL_W)
00087 #define CAR_H 40
00088 #define CAR_OFFSET_X 0
00089 #define CAR_OFFSET_Y (CELL_H / 2 - CAR_H / 2)
00090
00091 // Tortugas
00092 #define TURTLE_FRAMES 11 // 11 frames distintos tiene la animación completa
00093 #define TURTLE_SIDE CELL_W
```

```
00094 #define TURTLE_FRAME_OFFSET_XY (CELL_W / 2 - TURTLE_SIDE / 2)
00095
00096 // Corazon (vidas)
00097 #define SPRITE_SIZE_HEART 25 // cuadrado
00098
00099 #define SPRITE_DEAD_SIZE 35 // cuadrado
00100 #define SPRITE_DEAD_OFFSET (CELL_W / 2 - SPRITE_DEAD_SIZE / 2)
00101
00102 #define SPRITE_COIN_FRAMES 6
00103 #define SPRITE_COIN_SIDE 24
00104 #define SPRITE_COIN_OFFSET_XY (CELL_W / 2 - SPRITE_COIN_SIDE / 2)
00105
00106 #define SPRITE_SPLASH_FRAMES 6
00107 #define SPRITE_SPLASH_W 98
00108 #define SPRITE_SPLASH_H 68
00109 #define SPRITE_SPLASH_OFFSET_X (CELL_W / 2 - SPRITE_SPLASH_W / 2)
00110 #define SPRITE_SPLASH_OFFSET_Y (CELL_W / 2 - SPRITE_SPLASH_H / 2)
00111
00112 #define SPRITE_BORDER_START_X 0
00113 #define SPRITE_BORDER_START_Y CELL_H
00114
00115 #define MENU_OPTION_TOPLEFT_X 45
00116 #define MENU_OPTION_TOPLEFT_Y 72
00117 #define MENU_OPTION_DELTA_Y 100
00118 #define MENU_OPTION_W 600
00119 #define MENU_OPTION_H 75
00120
00121 #define CREDITS_SCREEN_LENGTH 2576
00122 #define CREDITS_SCREEN_START 0
00123 #define CREDITS_SCREEN_FINAL (CREDITS_SCREEN_LENGTH - DISPLAY_H)
00124
00125 // Factor que determina cuando considerar que un bloque esta dentro de otro (ver
      'inside_short_scaled')
00126 #define INSERTION_FACTOR (double)0.5
00127
00128 #define GOAL_ROW_OFFSET_Y_FIX 5    // baja un poco mas en Y
00129 #define GOAL_ROW_MARGIN_TO_REACH 5 // holgura para meterse a uno de los goals
00130
00131 /*******************************************************************************
00132  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00133  *******************************************************************************/
00134
00135 typedef struct
00136 {
00137     int x;
00138     int y;
00139
00140 } pair_xy_t;
00141
00142 enum GOALS
00143 {
00144     GOAL_LEFT,
00145     GOAL_LEFT_MID,
00146     GOAL_MID,
00147     GOAL_RIGHT_MID,
00148     GOAL_RIGHT,
00149     MAX_GOALS
00150 };
00151
00152 enum DIRECTIONS
00153 {
00154     DIRECTION_NONE,
00155     DIRECTION_UP,
00156     DIRECTION_RIGHT,
00157     DIRECTION_LEFT,
00158     DIRECTION_DOWN
00159 };
00160
00161 enum MENU_STATES
00162 {
00163     MENU_STATE_OPCION_0,
00164     MENU_STATE_OPCION_1,
00165     MENU_STATE_OPCION_2,
00166     MENU_STATE_OPCION_3,
00167     MENU_STATE_OPCION_4,
00168     MENU_STATE_MAX
00169 };
00170
00171 enum MENU_WINDOWS
00172 {
00173     MENU_WINDOW_HOME,
00174     MENU_WINDOW_DIFFICULTY,
00175     MENU_WINDOW_PAUSE,
00176     MENU_WINDOW_GAME_OVER,
00177     MENU_WINDOW_MAX
00178 };
00179
```

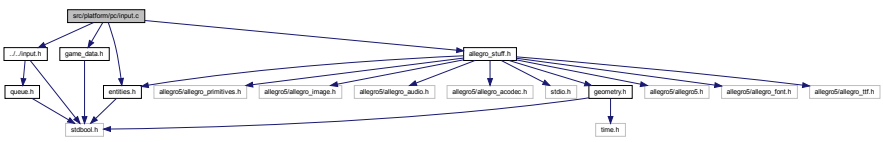
```
00180 typedef enum CAR_TYPE
00181 {
00182     CAR_BLUE = 0,
00183     CAR_POLICE,
00184     CAR_YELLOW,
00185     TRUCK_FIRE,
00186     TRUCK,
00187     CAR_TYPE_N
00188 } CAR_TYPE;
00189
00190 /*****************************************************************************
00191  * VARIABLE PROTOTYPES WITH GLOBAL SCOPE
00192  *****************************************************************************/
00193
00198 extern const unsigned int lanes_logs[LANES_LOG_TOTAL];
00199
00204 extern const unsigned int lanes_turtles[LANES_TURTLE_TOTAL];
00205
00210 extern const unsigned int lanes_cars[LANES_CAR_TOTAL];
00211
00216 extern const unsigned int goal_cols[MAX_GOALS];
00217
00218 /*****************************************************************************
00219  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00220  *****************************************************************************/
00221
00229 int get_rand_between(int low, int high);
00230
00245 bool collide(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2);
00246
00261 bool collideShort(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh);
00262
00277 bool inside(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2);
00278
00293 bool insideShort(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh);
00294
00310 bool insideShortScaled(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh, float scale);
00311
00324 int mapInt(int source, int min_in, int max_in, int min_out, int max_out);
00325
00332 pair_xy_t getXYFromFrogFrame(int frame);
00333
00340 pair_xy_t getXYFromTurtleFrame(int frame);
00341
00348 pair_xy_t getXYFromCarFrame(int frame);
00349
00356 pair_xy_t getXYFromCoinFrame(int frame);
00357
00364 pair_xy_t getXYFromSplashFrame(int frame);
00365
00366 /*****************************************************************************
00367  *****************************************************************************/
00368
00369 #endif // _GEOMETRY_H_
```

4.39 src/platform/pc/input.c File Reference

Source del modulo input, orientado a PC. Se encarga de procesar las entradas en la implementación de PC, y devolverlas adecuadamente a la FSM.

```
#include "../../input.h"
#include "allegro_stuff.h"
#include "entities.h"
#include "game_data.h"
```

Include dependency graph for input.c:



Functions

- void iniciarEntradas (void)

  Inicializa las entradas de la plataforma.
- event_t leerEntradas (void)

  Devuelve una entrada válida.

4.39.1 Detailed Description

Source del modulo input, orientado a PC. Se encarga de procesar las entradas en la implementación de PC, y devolverlas adecuadamente a la FSM.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file input.c.

4.39.2 Function Documentation

4.39.2.1 iniciarEntradas() `void iniciarEntradas ( void )`

Inicializa las entradas de la plataforma.

Definition at line 37 of file input.c.

4.39.2.2 leerEntradas() `event_t leerEntradas ( void )`

Devuelve una entrada válida.

Returns

event_t enum eventos_tecla

Definition at line 41 of file input.c.

4.40 input.c

Go to the documentation of this file.

```
00001
00013 /*****************************************************************************
00014  * INCLUDE HEADER FILES
00015  *****************************************************************************/
00016
00017 #include "../../input.h"
00018
00019 #include "allegro_stuff.h"
00020 #include "entities.h"
00021 #include "game_data.h"
00022
00023 /*****************************************************************************
00024  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00025  *****************************************************************************/
00026
00027 static ALLEGRO_EVENT *event;
00028
00029 static unsigned char last_key;
00030
00031 /*****************************************************************************
00032  *****************************************************************************
00033                          GLOBAL FUNCTION DEFINITIONS
00034  *****************************************************************************
00035  *****************************************************************************/
00036
00037 void iniciarEntradas(void)
00038 {
00039 }
00040
00041 event_t leerEntradas(void)
00042 {
00043     event_t retorno = NO_MOVER;
00044     // bool queue_no_empty;
00045
00046     event = allegro_get_next_event();
00047     int i;
00048
00049     if (event != NULL)
00050     {
00051         switch ((*event).type)
00052         {
00053         case ALLEGRO_EVENT_TIMER:
00054             allegro_set_var_redraw(true);
00055             break;
00056
00057         case ALLEGRO_EVENT_KEY_DOWN:
00058             last_key = allegro_get_last_key();
00059
00060             if (last_key != (*event).keyboard.keycode)
00061             {
00062                 retorno = (*event).keyboard.keycode;
00063                 allegro_set_last_key(retorno);
00064
00065                 switch (retorno)
00066                 {
00067                 case ALLEGRO_KEY_F2:
00068                     allegro_sound_set_stream_gain_up();
00069                     break;
00070
00071                 case ALLEGRO_KEY_F1:
00072                     allegro_sound_set_stream_gain_down();
00073                     break;
00074
00075                 case ALLEGRO_KEY_8:
00076                     break;
00077
00078                 case ALLEGRO_KEY_9:
00079                     break;
00080
00081                 case ALLEGRO_KEY_1:
00082                     game_data_add_score();
00083                     break;
00084
00085                 case ALLEGRO_KEY_2:
00086                     retorno = GAME_OVER;
00087                     break;
00088
00089                 case ALLEGRO_KEY_3:
00090                     // int i;
00091                     for (i = 0; i < MAX_GOALS; i++)
00092                         game_data_set_goal(i);
00093                     break;
00094
```

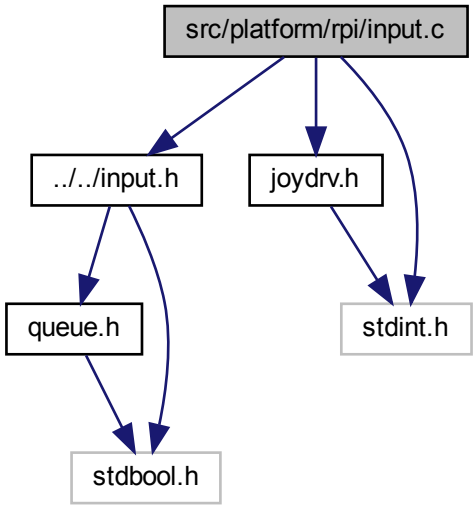
```
00095                 case ALLEGRO_KEY_4:
00096                     game_data_reset_goals();
00097                     break;
00098 
00099                 case ALLEGRO_KEY_5:
00100                     game_data_add_run_time_goal();
00101                     break;
00102 
00103                 default:
00104                     break;
00105                 }
00106             }
00107 
00108             break;
00109 
00110         case ALLEGRO_EVENT_KEY_UP:
00111             allegro_set_last_key(0);
00112 
00113             break;
00114 
00115         case ALLEGRO_EVENT_DISPLAY_CLOSE:
00116             retorno = FORCE_SALIR;
00117             break;
00118 
00119         default:
00120             break;
00121         }
00122     }
00123 
00124     return retorno;
00125 }
```

4.41 src/platform/rpi/input.c File Reference

Archivo para manejo del joystick en RPI.

```
#include "../../input.h"
#include "joydrv.h"
#include <stdint.h>
```

Include dependency graph for input.c:



Functions

- void iniciarEntradas ()

  Inicializa las entradas de la plataforma.
- event_t leerEntradas ()

  Devuelve una entrada válida.

4.41.1 Detailed Description

Archivo para manejo del joystick en RPI.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file input.c.

4.41.2 Function Documentation

4.41.2.1 iniciarEntradas() `void iniciarEntradas ( void )`

Inicializa las entradas de la plataforma.

Definition at line 41 of file input.c.

4.41.2.2 leerEntradas() `event_t leerEntradas ( void )`

Devuelve una entrada válida.

Returns

event_t enum eventos_tecla

Definition at line 46 of file input.c.

4.42 input.c

Go to the documentation of this file.

```
00001
00008 /*****************************************************************************
00009  * INCLUDE HEADER FILES
00010  *****************************************************************************/
00011
00012 #include "../../input.h"
00013 #include "joydrv.h"
00014
00015 #include <stdint.h>
00016
00017 /*****************************************************************************
00018  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00019  *****************************************************************************/
00020
00027 static int8_t modulo(int8_t x);
00028
00029 /*****************************************************************************
00030  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00031  *****************************************************************************/
00032
00033 static int prev = NO_MOVER; // estado previo del joystick
00034
00035 /*****************************************************************************
00036  *****************************************************************************
00037                         GLOBAL FUNCTION DEFINITIONS
00038  *****************************************************************************
00039  *****************************************************************************/
00040
00041 void iniciarEntradas()
00042 {
00043     joy_init();
00044 }
00045
00046 event_t leerEntradas()
00047 {
00048     joy_update();
00049     int act = joy_get_switch();
00050     if (act == J_PRESS)
00051     {
00052         if (prev != J_PRESS)
00053         {
00054             prev = J_PRESS;
00055             return ENTER;
00056         }
00057         return NO_MOVER;
00058     }
00059
00060     jcoord_t crd = joy_get_coord();
00061
00062     int umbral;
00063
00064     if (prev == NO_MOVER)
00065         umbral = 20;
00066     else
00067         umbral = 10;
00068
00069     if ((crd.y - umbral) > modulo(crd.x))
00070         act = ARRIBA;
00071     else if ((crd.y + umbral) < -(modulo(crd.x)))
00072         act = ABAJO;
00073     else if ((crd.x - umbral) > modulo(crd.y))
00074         act = DCHA;
00075     else if ((crd.x + umbral) < -(modulo(crd.y)))
00076         act = IZDA;
00077     else
00078         act = NO_MOVER;
00079
00080     if (act != prev)
00081     {
00082         prev = act;
00083         return act;
00084     }
00085
00086     return NO_MOVER;
00087 }
00088
00089 /*****************************************************************************
00090  *****************************************************************************
00091                         LOCAL FUNCTION DEFINITIONS
00092  *****************************************************************************
00093  *****************************************************************************/
00094
00095 static int8_t modulo(int8_t x)
```

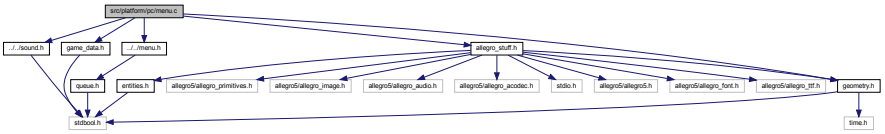
```
00096 {
00097     if (x == -128)
00098         return 127; // excepción: caso en el cual -(-128) = -128
00099
00100     return x >= 0 ? x : -x;
00101 }
```

4.43 src/platform/pc/menu.c File Reference

Source del modulo menu, orientado a PC. Se encarga de la inicialización, actualización y muestreo de lo relacionado al menú en PC. Funciones llamadas por la FSM.

```
#include "../../menu.h"
#include "../../sound.h"
#include "allegro_stuff.h"
#include "geometry.h"
#include "game_data.h"
```

Include dependency graph for menu.c:



Data Structures

- struct window_t
- struct menu_t

Macros

- #define STATS_X_COORD 20
- #define STATS_Y_COORD_START (DISPLAY_H / 2 + 50)

Functions

- void iniciarMenu (void)

  Inicia el menu.
- void destruirMenu (void)

  Destruye del menu.
- void setMenu (int *a, unsigned int size)

  Selecciona un menu.
- void setOpcion (int opc)

  Selecciona una opcion del menu.
- int getOpcion (void)

  Devuelve la opcion actual del menu.
- void subirOpcion (void)

  Selecciona la opcion superior a la actual.
- void bajarOpcion (void)

  Selecciona la opcion inferior a la actual.
- void moverOpcionActual (void)

4.43.1 Detailed Description

Source del modulo menu, orientado a PC. Se encarga de la inicialización, actualización y muestreo de lo relacionado al menú en PC. Funciones llamadas por la FSM.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file menu.c.

4.43.2 Macro Definition Documentation

4.43.2.1 STATS_X_COORD `#define STATS_X_COORD 20`

Definition at line 28 of file menu.c.

4.43.2.2 STATS_Y_COORD_START `#define STATS_Y_COORD_START (DISPLAY_H / 2 + 50)`

Definition at line 29 of file menu.c.

4.43.3 Function Documentation

4.43.3.1 bajarOpcion() `void bajarOpcion ( void )`

Selecciona la opcion inferior a la actual.

Definition at line 155 of file menu.c.

4.43.3.2 destruirMenu() `void destruirMenu ( void )`

Destruye del menu.

Definition at line 88 of file menu.c.

4.43.3.3 getOpcion() `int getOpcion ( void )`

Devuelve la opcion actual del menu.

Returns

int Opcion seleccionada actualmente

Definition at line 135 of file menu.c.

4.43.3.4 iniciarMenu() `void iniciarMenu ( void )`

Inicia el menu.

Definition at line 83 of file menu.c.

4.43.3.5 moverOpcionActual() `void moverOpcionActual ( void )`

Definition at line 170 of file menu.c.

4.43.3.6 setMenu() `void setMenu ( int * a, unsigned int size )`

Selecciona un menu.

Parameters

a	Puntero a textos del menu
size	Opciones del menu

Definition at line 93 of file menu.c.

4.43.3.7 setOpcion() `void setOpcion ( int opc )`

Selecciona una opcion del menu.

Parameters

opc	Opcion a seleccionar

Definition at line 127 of file menu.c.

4.43.3.8 subirOpcion() void subirOpcion (void)

Selecciona la opcion superior a la actual.

Definition at line 140 of file menu.c.

4.44 menu.c

Go to the documentation of this file.

```
00001
00013 /**********************************************************************************
00014  * INCLUDE HEADER FILES
00015  **********************************************************************************/
00016
00017 #include "../../menu.h"
00018 #include "../../sound.h"
00019
00020 #include "allegro_stuff.h"
00021 #include "geometry.h"
00022 #include "game_data.h"
00023
00024 /**********************************************************************************
00025  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00026  **********************************************************************************/
00027
00028 #define STATS_X_COORD 20
00029 #define STATS_Y_COORD_START (DISPLAY_H / 2 + 50)
00030
00031 /**********************************************************************************
00032  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00033  **********************************************************************************/
00034
00035 typedef struct
00036 {
00037     int actual_state;
00038     int max_states;
00039 } window_t;
00040
00041 typedef struct
00042 {
00043     window_t window[MENU_WINDOW_MAX];
00044
00045     int actual_window;
00046
00047 } menu_t;
00048
00049 /**********************************************************************************
00050  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00051  **********************************************************************************/
00052
00057 static void inicializarMenu(void);
00058
00063 static void renderizarMenu(void);
00064
00069 static void show_stats(void);
00070
00071 /**********************************************************************************
00072  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00073  **********************************************************************************/
00074
00075 static menu_t menu;
00076
00077 /**********************************************************************************
```

```
00078  ******************************************************************************
00079                          GLOBAL FUNCTION DEFINITIONS
00080  ******************************************************************************
00081  ******************************************************************************/
00082
00083 void iniciarMenu(void)
00084 {
00085     inicializarMenu();
00086 }
00087
00088 void destruirMenu(void)
00089 {
00090     allegro_deinits();
00091 }
00092
00093 void setMenu(int *a, unsigned int size)
00094 {
00095     switch (a[0])
00096     {
00097     // menu principal (JUGAR, DIFICULTAD, RANKING, SALIRTXT)
00098     case JUGAR:
00099         menu.actual_window = MENU_WINDOW_HOME;
00100
00101         break;
00102
00103     // menu dificultades (FACIL, NORMAL, DIFICIL)
00104     case FACIL:
00105         menu.actual_window = MENU_WINDOW_DIFFICULTY;
00106
00107         break;
00108
00109     // menu pausa (CONTINUAR, REINICIAR, SALIRTXT)
00110     case CONTINUAR:
00111         menu.actual_window = MENU_WINDOW_PAUSE;
00112         allegro_set_rick_flag(false);
00113
00114         break;
00115
00116     // menu game over (REINICIAR, SALIRTXT)
00117     case REINICIAR:
00118         menu.actual_window = MENU_WINDOW_GAME_OVER;
00119
00120         break;
00121
00122     default:
00123         break;
00124     }
00125 }
00126
00127 void setOpcion(int opc)
00128 {
00129     // Seleccina uno de los botones del menu actual
00130     menu.window[menu.actual_window].actual_state = opc;
00131
00132     renderizarMenu();
00133 }
00134
00135 int getOpcion(void)
00136 {
00137     return (menu.window[menu.actual_window].actual_state);
00138 }
00139
00140 void subirOpcion(void)
00141 {
00142     int *actual_option = &menu.window[menu.actual_window].actual_state;
00143     int *max_option = &menu.window[menu.actual_window].max_states;
00144
00145     (*actual_option)--;
00146
00147     if (*actual_option < 0)
00148         setOpcion(*max_option - 1);
00149     else
00150         renderizarMenu();
00151
00152     reproducirEfecto(EFECTO_SELECCION);
00153 }
00154
00155 void bajarOpcion(void)
00156 {
00157     int *actual_option = &menu.window[menu.actual_window].actual_state;
00158     int *max_option = &menu.window[menu.actual_window].max_states;
00159
00160     (*actual_option)++;
00161
00162     if (*actual_option == *max_option)
00163         setOpcion(0);
00164     else
```

```
00165         renderizarMenu();
00166
00167     reproducirEfecto(EFECTO_SELECCION);
00168 }
00169
00170 void moverOpcionActual(void)
00171 {
00172 }
00173
00174 /*****************************************************************************
00175  *****************************************************************************
00176                         LOCAL FUNCTION DEFINITIONS
00177  *****************************************************************************
00178  *****************************************************************************/
00179
00180 static void inicializarMenu(void)
00181 {
00182     // menu principal (JUGAR, DIFICULTAD, RANKING, SALIRTXT)
00183     menu.window[MENU_WINDOW_HOME].max_states = 5;
00184
00185     // menu dificultades (FACIL, NORMAL, DIFICIL)
00186     menu.window[MENU_WINDOW_DIFFICULTY].max_states = 3;
00187
00188     // menu pausa (CONTINUAR, REINICIAR, SALIRTXT)
00189     menu.window[MENU_WINDOW_PAUSE].max_states = 3;
00190
00191     // menu game over (REINICIAR, SALIRTXT)
00192     menu.window[MENU_WINDOW_GAME_OVER].max_states = 2;
00193 }
00194
00195 static void renderizarMenu()
00196 {
00197     allegro_clear_display();
00198
00199     ALLEGRO_BITMAP *background = NULL;
00200     ALLEGRO_BITMAP *option = NULL;
00201
00202     background = sprites.menu[menu.actual_window].background;
00203     option = sprites.menu[menu.actual_window].option[menu.window[menu.actual_window].actual_state];
00204
00205     al_draw_bitmap(background, 0, 0, 0);
00206
00207     al_draw_bitmap(option, MENU_OPTION_TOPLEFT_X,
00208                    MENU_OPTION_TOPLEFT_Y + (menu.window[menu.actual_window].actual_state *
      MENU_OPTION_DELTA_Y),
00209                    0);
00210
00211     if (menu.actual_window == MENU_WINDOW_PAUSE || menu.actual_window == MENU_WINDOW_GAME_OVER)
00212         show_stats();
00213
00214     al_flip_display();
00215 }
00216
00217 static void show_stats(void)
00218 {
00219     char *name = game_data_get_name();
00220     int score = game_data_get_score();
00221     int max_score = game_data_get_old_max_score();
00222     ALLEGRO_COLOR color = al_map_rgb(255, 255, 255);
00223     ALLEGRO_FONT *font = allegro_get_var_font();
00224
00225     int x = STATS_X_COORD;
00226     int y = STATS_Y_COORD_START;
00227
00228     al_draw_textf(font, color,
00229                   x,
00230                   y,
00231                   0,
00232                   "Jugador: %s", name);
00233
00234     y += 20;
00235
00236     if (menu.actual_window == MENU_WINDOW_PAUSE)
00237     {
00238         al_draw_textf(font, color,
00239                       x,
00240                       y,
00241                       0,
00242                       "Score:   %d", score);
00243
00244         y += 20;
00245
00246         al_draw_textf(font, color,
00247                       x,
00248                       y,
00249                       0,
00250                       "Max score:   %d", max_score);
```

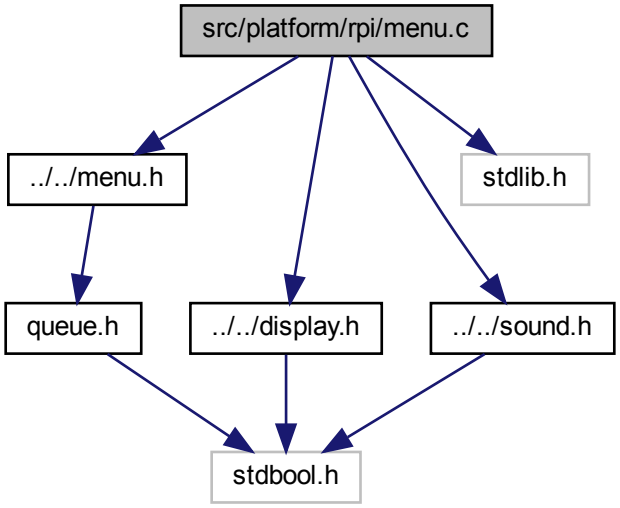
```
00251     }
00252     else if (menu.actual_window == MENU_WINDOW_GAME_OVER)
00253     {
00254         if (score > max_score)
00255         {
00256             al_draw_textf(font, color,
00257                           x,
00258                           y + 10,
00259                           0,
00260                           "NUEVA PUNTUACION MAXIMA!");
00261 
00262             y += 40;
00263 
00264             al_draw_textf(font, color,
00265                           x,
00266                           y,
00267                           0,
00268                           "Anterior: %d", max_score);
00269 
00270             y += 20;
00271 
00272             al_draw_textf(font, color,
00273                           x,
00274                           y,
00275                           0,
00276                           "Nueva: %d", score);
00277         }
00278 
00279         else
00280         {
00281             al_draw_textf(font, color,
00282                           x,
00283                           y,
00284                           0,
00285                           "Score:   %d", score);
00286 
00287             y += 20;
00288 
00289             al_draw_textf(font, color,
00290                           x,
00291                           y,
00292                           0,
00293                           "Max score:   %d", max_score);
00294         }
00295     }
00296 }
```

4.45 src/platform/rpi/menu.c File Reference

Archivo para manejo de los menús en RPI.

```
#include "../../menu.h"
#include "../../display.h"
#include "../../sound.h"
#include <stdlib.h>
```

Include dependency graph for menu.c:



Functions

- void setMenu (int *a, unsigned int size)
  Selecciona un menu.
- void setOpcion (int opc)
  Selecciona una opcion del menu.
- int getOpcion ()
  Devuelve la opcion actual del menu.
- void subirOpcion ()
  Selecciona la opcion superior a la actual.
- void bajarOpcion ()
  Selecciona la opcion inferior a la actual.
- void iniciarMenu ()
  Inicia el menu.
- void destruirMenu ()
  Destruye del menu.

4.45.1 Detailed Description

Archivo para manejo de los menús en RPI.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file menu.c.

4.45.2 Function Documentation

4.45.2.1 bajarOpcion() `void bajarOpcion ( void )`

Selecciona la opcion inferior a la actual.

Definition at line 79 of file menu.c.

4.45.2.2 destruirMenu() `void destruirMenu ( void )`

Destruye del menu.

Definition at line 92 of file menu.c.

4.45.2.3 getOpcion() `int getOpcion ( void )`

Devuelve la opcion actual del menu.

Returns

int Opcion seleccionada actualmente

Definition at line 65 of file menu.c.

4.45.2.4 iniciarMenu() `void iniciarMenu ( void )`

Inicia el menu.

Definition at line 88 of file menu.c.

4.45.2.5 setMenu() `void setMenu ( int * a, unsigned int size )`

Selecciona un menu.

Parameters

a	Puntero a textos del menu
size	Opciones del menu

Definition at line 41 of file menu.c.

4.45.2.6 setOpcion() `void setOpcion ( int opc )`

Selecciona una opcion del menu.

Parameters

opc	Opcion a seleccionar

Definition at line 59 of file menu.c.

4.45.2.7 subirOpcion() `void subirOpcion ( void )`

Selecciona la opcion superior a la actual.

Definition at line 70 of file menu.c.

4.45.3 Variable Documentation

4.45.3.1 max_opciones `int max_opciones`

Definition at line 26 of file menu.c.

4.45.3.2 menu_actual `int* menu_actual`

Definition at line 24 of file menu.c.

4.45.3.3 opcion_actual `int opcion_actual`

Definition at line 25 of file menu.c.

4.46 menu.c

Go to the documentation of this file.

```
00001
00008 /*****************************************************************************
00009  * INCLUDE HEADER FILES
00010  *****************************************************************************/
00011
00012 #include "../../menu.h"
00013 #include "../../display.h"
00014 #include "../../sound.h"
00015
00016 #include <stdlib.h>
00017
00018 /*****************************************************************************
00019  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00020  *****************************************************************************/
00021
00022 static struct
00023 {
00024     int *menu_actual; // arreglo con los índices de textos ordenados para mostrar como menú
00025     int opcion_actual;
00026     int max_opciones;
00027 } menu;
00028
00029 /*****************************************************************************
00030  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00031  *****************************************************************************/
00032
00033 static char *menu_textos[] = {"JUGAR", "DIFICULTAD", "RANKING", "CREDITOS", "SALIR", "CONTINUAR",
       "REINICIAR", "FACIL", "NORMAL", "DIFICIL"};
00034
00035 /*****************************************************************************
00036  *****************************************************************************
00037                         GLOBAL FUNCTION DEFINITIONS
00038  *****************************************************************************
00039  *****************************************************************************/
00040
00041 void setMenu(int *a, unsigned int size)
00042 {
00043     int *aux = realloc(menu.menu_actual, size * sizeof(int));
00044     if (aux == NULL)
00045     {
00046         free(menu.menu_actual);
00047         queueInsertar(FORCE_SALIR);
00048     }
00049
00050     menu.menu_actual = aux;
00051     int i;
00052     for (i = 0; i < size; i++)
00053     {
00054         menu.menu_actual[i] = a[i];
00055     }
00056     menu.max_opciones = size;
00057 }
00058
00059 void setOpcion(int opc)
00060 {
00061     menu.opcion_actual = opc;
00062     dejarTexto(menu_textos[menu.menu_actual[menu.opcion_actual]], POS_OPCION, true);
00063 }
00064
00065 int getOpcion()
00066 {
00067     return menu.opcion_actual;
00068 }
00069
00070 void subirOpcion()
00071 {
00072     if (--menu.opcion_actual < 0)
00073         menu.opcion_actual = menu.max_opciones - 1;
00074     dejarTexto(menu_textos[menu.menu_actual[menu.opcion_actual]], POS_OPCION, true);
00075
00076     reproducirEfecto(EFECTO_SELECCION);
00077 }
00078
00079 void bajarOpcion()
00080 {
00081     if (++menu.opcion_actual >= menu.max_opciones)
00082         menu.opcion_actual = 0;
00083     dejarTexto(menu_textos[menu.menu_actual[menu.opcion_actual]], POS_OPCION, true);
00084
00085     reproducirEfecto(EFECTO_SELECCION);
00086 }
00087
00088 void iniciarMenu()
```

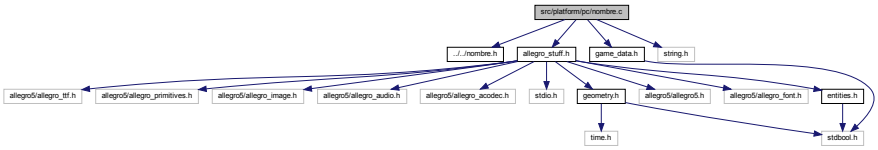
```
00089 {
00090 }
00091
00092 void destruirMenu()
00093 {
00094     free(menu.menu_actual);
00095 }
```

4.47 src/platform/pc/nombre.c File Reference

Source del modulo nombre, orientado a PC. Se encarga del manejo del nombre del jugador, teniendo funciones que llama la FSM.

```
#include "../../nombre.h"
#include "allegro_stuff.h"
#include "game_data.h"
#include <string.h>
```

Include dependency graph for nombre.c:



Macros

- #define NAME_TOPLEFT_X 55
- #define NAME_TOPLEFT_Y 312

Functions

- void nuevoNombre (void)

  Se ejecuta una vez al ingresar a poner un nuevo nombre.
- void subirLetra (void)

  Selecciona la siguiente letra superior.
- void bajarLetra (void)

  Selecciona la letra inferior.
- void siguienteLetra (void)

  Confirma la letra y pasa a seleccionar la siguiente.
- void agregarLetra (void)

  Confirma la letra.
- void subirNombre (void)
- char * devolverNombre (void)

  Devuelve puntero al nombre.

4.47.1 Detailed Description

Source del modulo nombre, orientado a PC. Se encarga del manejo del nombre del jugador, teniendo funciones que llama la FSM.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file nombre.c.

4.47.2 Macro Definition Documentation

4.47.2.1 NAME_TOPLEFT_X `#define NAME_TOPLEFT_X 55`

Definition at line 27 of file nombre.c.

4.47.2.2 NAME_TOPLEFT_Y `#define NAME_TOPLEFT_Y 312`

Definition at line 28 of file nombre.c.

4.47.3 Function Documentation

4.47.3.1 agregarLetra() `void agregarLetra ( void )`

Confirma la letra.

Definition at line 61 of file nombre.c.

4.47.3.2 bajarLetra() `void bajarLetra ( void )`

Selecciona la letra inferior.

Definition at line 53 of file nombre.c.

4.47.3.3 devolverNombre() `char * devolverNombre ( void )`

Devuelve puntero al nombre.

Returns

char* Puntero al nombre

Definition at line 82 of file nombre.c.

4.47.3.4 nuevoNombre() `void nuevoNombre ( void )`

Se ejecuta una vez al ingresar a poner un nuevo nombre.

Definition at line 36 of file nombre.c.

4.47.3.5 siguienteLetra() `void siguienteLetra ( void )`

Confirma la letra y pasa a seleccionar la siguiente.

Definition at line 57 of file nombre.c.

4.47.3.6 subirLetra() `void subirLetra ( void )`

Selecciona la siguiente letra superior.

Definition at line 49 of file nombre.c.

4.47.3.7 subirNombre() `void subirNombre ( void )`

Definition at line 78 of file nombre.c.

4.48 nombre.c

Go to the documentation of this file.

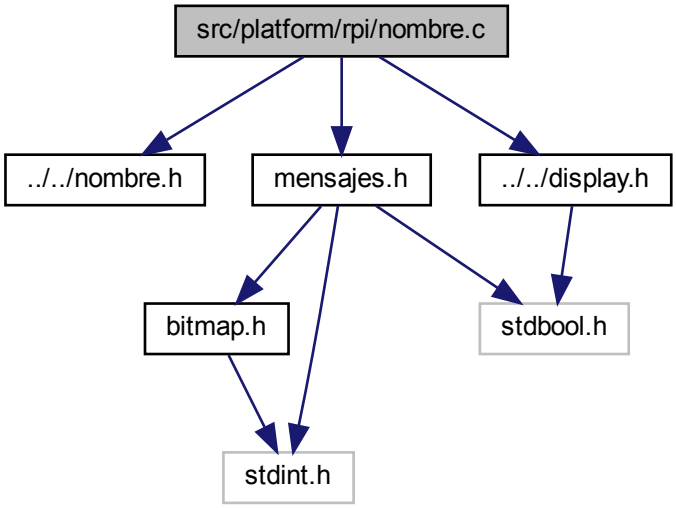
```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "../../nombre.h"
00017
00018 #include "allegro_stuff.h"
00019 #include "game_data.h"
00020
00021 #include <string.h>
00022
00023 /*******************************************************************************
00024  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00025  ******************************************************************************/
00026
00027 #define NAME_TOPLEFT_X 55
00028 #define NAME_TOPLEFT_Y 312
00029
00030 /*******************************************************************************
00031  *******************************************************************************
00032                         GLOBAL FUNCTION DEFINITIONS
00033  *******************************************************************************
00034  ******************************************************************************/
00035
00036 void nuevoNombre(void)
00037 {
00038     allegro_clear_display();
00039
00040     game_data_clear_name();
00041     game_data_set_score_max(0);
00042
00043     /*cambiar por background correspondiente*/
00044     al_draw_bitmap(sprites.name, 0, 0, 0);
00045
00046     al_flip_display();
00047 }
00048
00049 void subirLetra(void)
00050 {
00051 }
00052
00053 void bajarLetra(void)
00054 {
00055 }
00056
00057 void siguienteLetra(void)
00058 {
00059 }
00060
00061 void agregarLetra(void)
00062 {
00063     game_data_add_name_letter(allegro_get_last_key());
00064
00065     char *name = game_data_get_name();
00066
00067     allegro_clear_display();
00068
00069     /*cambiar por background correspondiente*/
00070     al_draw_bitmap(sprites.name, 0, 0, 0);
00071
00072     al_draw_textf(allegro_get_var_font(), al_map_rgb(100, 200, 200), NAME_TOPLEFT_X, NAME_TOPLEFT_Y,
      0,
00073                   "%s", name);
00074
00075     al_flip_display();
00076 }
00077
00078 void subirNombre(void)
00079 {
00080 }
00081
00082 char *devolverNombre(void)
00083 {
00084     return game_data_get_name();
00085 }
```

4.49 src/platform/rpi/nombre.c File Reference

Archivo para manejo de información en el ingreso del nombre.

```
#include "../../nombre.h"
#include "mensajes.h"
#include "../../display.h"
```

Include dependency graph for nombre.c:



Functions

- void nuevoNombre ()

  Se ejecuta una vez al ingresar a poner un nuevo nombre.
- void subirLetra ()

  Selecciona la siguiente letra superior.
- void bajarLetra ()

  Selecciona la letra inferior.
- void siguienteLetra ()

  Confirma la letra y pasa a seleccionar la siguiente.
- void agregarLetra (void)

  Confirma la letra.
- char * devolverNombre (void)

  Devuelve puntero al nombre.

Variables

- matriz_t disp_matriz

4.49.1 Detailed Description

Archivo para manejo de información en el ingreso del nombre.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file nombre.c.

4.49.2 Function Documentation

4.49.2.1 agregarLetra() `void agregarLetra ( void )`

Confirma la letra.

Definition at line 71 of file nombre.c.

4.49.2.2 bajarLetra() `void bajarLetra ( void )`

Selecciona la letra inferior.

Definition at line 54 of file nombre.c.

4.49.2.3 devolverNombre() `char * devolverNombre ( void )`

Devuelve puntero al nombre.

Returns

char∗ Puntero al nombre

Definition at line 75 of file nombre.c.

4.49.2.4 nuevoNombre() void nuevoNombre (void)

Se ejecuta una vez al ingresar a poner un nuevo nombre.

Definition at line 35 of file nombre.c.

4.49.2.5 siguienteLetra() void siguienteLetra (void)

Confirma la letra y pasa a seleccionar la siguiente.

Definition at line 63 of file nombre.c.

4.49.2.6 subirLetra() void subirLetra (void)

Selecciona la siguiente letra superior.

Definition at line 45 of file nombre.c.

4.49.3 Variable Documentation

4.49.3.1 disp_matriz matriz_t disp_matriz [extern]

Definition at line 37 of file display.c.

4.50 nombre.c

Go to the documentation of this file.

```
00001
00008 /*****************************************************************************
00009  * INCLUDE HEADER FILES
00010  *****************************************************************************/
00011
00012 #include "../../nombre.h"
00013 #include "mensajes.h"
00014 #include "../../display.h"
00015
00016 /*****************************************************************************
00017  * VARIABLES WITH GLOBAL SCOPE
00018  *****************************************************************************/
00019
00020 extern matriz_t disp_matriz;
00021
00022 /*****************************************************************************
00023  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00024  *****************************************************************************/
00025
00026 static mensaje_t nombre;
00027 static char last;
00028
00029 /*****************************************************************************
```

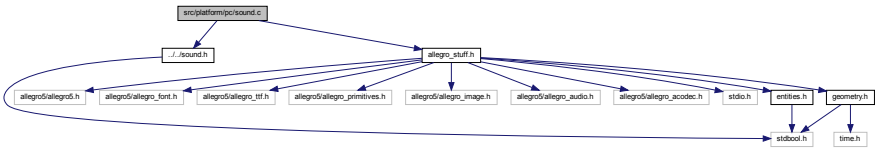
```
00030  ********************************************************************************
00031             GLOBAL FUNCTION DEFINITIONS
00032  ********************************************************************************
00033  ********************************************************************************/
00034
00035 void nuevoNombre()
00036 {
00037     nombre = mensaje("A", POS_MSJ2, false);
00038     nombre.index = 0;
00039     nombre.j = 0;
00040     last = 'A';
00041     copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00042     actualizarDisplay();
00043 }
00044
00045 void subirLetra()
00046 {
00047     if (--last < 'A')
00048         last = 'Z';
00049     reemplazarUltLetraMensaje(last, &nombre);
00050     copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00051     actualizarDisplay();
00052 }
00053
00054 void bajarLetra()
00055 {
00056     if (++last > 'Z')
00057         last = 'A';
00058     reemplazarUltLetraMensaje(last, &nombre);
00059     copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00060     actualizarDisplay();
00061 }
00062
00063 void siguienteLetra()
00064 {
00065     concatenarLetraMensaje(last, &nombre);
00066     last = 'A';
00067     copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00068     actualizarDisplay();
00069 }
00070
00071 void agregarLetra(void)
00072 {
00073 }
00074
00075 char *devolverNombre(void)
00076 {
00077     return nombre.msj;
00078 }
```

4.51 src/platform/pc/sound.c File Reference

Source del modulo sound. orientado a PC. Se encarga iniciar y reproducir efectos y musicas en la implementación de PC, cuando la FSM lo indique.

```
#include "../../sound.h"
#include "allegro_stuff.h"
```

Include dependency graph for sound.c:



Functions

- bool iniciarSonido (void)

  Inicializa el sonido de la plataforma.

- void destruirSonido (void)

  Desinicializa el sonido de la plataforma.
- void pausarMusica (void)

  Pausa la musica actual.
- void reproducirMusica (int musica)

  Pone a reproducir una musica dada.
- void reproducirEfecto (int sonido)

  Pone a reproducir un efecto dado.

4.51.1 Detailed Description

Source del modulo sound. orientado a PC. Se encarga iniciar y reproducir efectos y musicas en la implementación de PC, cuando la FSM lo indique.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file sound.c.

4.51.2 Function Documentation

4.51.2.1 destruirSonido() `void destruirSonido ( void )`

Desinicializa el sonido de la plataforma.

Definition at line 32 of file sound.c.

4.51.2.2 iniciarSonido() `bool iniciarSonido ( void )`

Inicializa el sonido de la plataforma.

Returns

true Exito

false Error

Definition at line 27 of file sound.c.

4.51.2.3 pausarMusica() void pausarMusica (void)

Pausa la musica actual.

Definition at line 36 of file sound.c.

4.51.2.4 reproducirEfecto() void reproducirEfecto (int *efecto*)

Pone a reproducir un efecto dado.

Parameters

int	num efectos

Definition at line 76 of file sound.c.

4.51.2.5 reproducirMusica() void reproducirMusica (int *musica*)

Pone a reproducir una musica dada.

Parameters

int	enum musica

Definition at line 41 of file sound.c.

4.52 sound.c

Go to the documentation of this file.

```
00001
00013 /*****************************************************************************
00014  * INCLUDE HEADER FILES
00015  *****************************************************************************/
00016
00017 #include "../../sound.h"
00018
00019 #include "allegro_stuff.h"
00020
00021 /*****************************************************************************
00022  *****************************************************************************
00023                         GLOBAL FUNCTION DEFINITIONS
00024  *****************************************************************************
00025  *****************************************************************************/
00026
00027 bool iniciarSonido(void)
00028 {
00029     return true;
00030 }
00031
00032 void destruirSonido(void)
00033 {
```

```
00034 }
00035
00036 void pausarMusica(void)
00037 {
00038     allegro_sound_pause_stream();
00039 }
00040
00041 void reproducirMusica(int musica)
00042 {
00043     switch (musica)
00044     {
00045     case MUSICA_CREDITOS:
00046         allegro_sound_set_stream_credits();
00047         break;
00048
00049     case MUSICA_JUGANDO:
00050         allegro_sound_set_stream_playing();
00051         break;
00052
00053     case MUSICA_MENU_PAUSA:
00054         allegro_sound_set_stream_pause_menu();
00055         break;
00056
00057     case MUSICA_MENU_PPAL:
00058         allegro_sound_set_stream_main_menu();
00059         break;
00060
00061     case MUSICA_RANKING:
00062         allegro_sound_set_stream_ranking();
00063         break;
00064
00065     case MUSICA_GAME_OVER:
00066         allegro_sound_set_stream_game_over();
00067         break;
00068
00069     default:
00070         break;
00071     }
00072
00073     allegro_sound_play_stream();
00074 }
00075
00076 void reproducirEfecto(int sonido)
00077 {
00078     switch (sonido)
00079     {
00080     case EFECTO_AHOGADO:
00081         allegro_sound_play_effect_drowned();
00082         break;
00083
00084     case EFECTO_IMPACTO:
00085         allegro_sound_play_effect_crash();
00086         break;
00087
00088     case EFECTO_MENU_ENTER:
00089         allegro_sound_play_effect_menu_enter();
00090         break;
00091
00092     case EFECTO_META:
00093         allegro_sound_play_effect_goal();
00094         break;
00095
00096     case EFECTO_NIVEL_COMPLETO:
00097         allegro_sound_play_effect_run_completed();
00098         break;
00099
00100     case EFECTO_NUEVO_MAX_SCORE:
00101         allegro_sound_play_effect_new_max_score();
00102         break;
00103
00104     case EFECTO_POCO_TIEMPO:
00105         allegro_sound_play_effect_low_time();
00106         break;
00107
00108     case EFECTO_SALIENDO:
00109         allegro_sound_play_effect_exiting();
00110         break;
00111
00112     case EFECTO_SALTO:
00113         allegro_sound_play_effect_jump();
00114         break;
00115
00116     case EFECTO_SELECCION:
00117         allegro_sound_play_effect_click();
00118         break;
00119
00120     default:
```

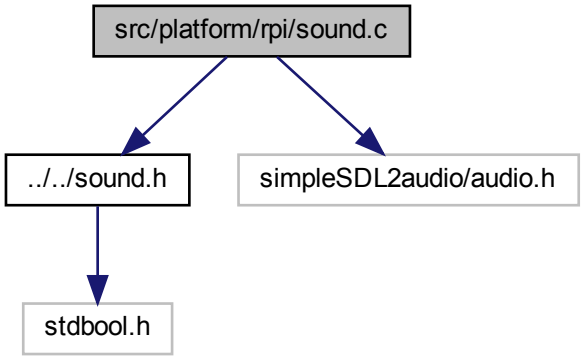
```
00121         break;
00122     }
00123 }
```

4.53 src/platform/rpi/sound.c File Reference

Archivo para manejo del sonido en RPI.

```
#include "../../sound.h"
#include "simpleSDL2audio/audio.h"
```

Include dependency graph for sound.c:



Macros

- #define MUSICA_DIR "../res/sounds/streams"
- #define EFECTOS_DIR "../res/sounds/samples"

Functions

- bool iniciarSonido (void)

  Inicializa el sonido de la plataforma.
- void destruirSonido (void)

  Desinicializa el sonido de la plataforma.
- void pausarMusica (void)

  Pausa la musica actual.
- void reproducirMusica (int m)

  Pone a reproducir una musica dada.
- void reproducirEfecto (int e)

  Pone a reproducir un efecto dado.

4.53.1 Detailed Description

Archivo para manejo del sonido en RPI.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file sound.c.

4.53.2 Macro Definition Documentation

4.53.2.1 EFECTOS_DIR `#define EFECTOS_DIR "../res/sounds/samples"`

Definition at line 20 of file sound.c.

4.53.2.2 MUSICA_DIR `#define MUSICA_DIR "../res/sounds/streams"`

Definition at line 19 of file sound.c.

4.53.3 Function Documentation

4.53.3.1 destruirSonido() `void destruirSonido ( void )`

Desinicializa el sonido de la plataforma.

Definition at line 69 of file sound.c.

4.53.3.2 iniciarSonido() `bool iniciarSonido ( void )`

Inicializa el sonido de la plataforma.

Returns

true Exito

false Error

Definition at line 57 of file sound.c.

4.53.3.3 pausarMusica() `void pausarMusica ( void )`

Pausa la musica actual.

Definition at line 76 of file sound.c.

4.53.3.4 reproducirEfecto() `void reproducirEfecto ( int efecto )`

Pone a reproducir un efecto dado.

Parameters

int	num efectos

Definition at line 95 of file sound.c.

4.53.3.5 reproducirMusica() `void reproducirMusica ( int musica )`

Pone a reproducir una musica dada.

Parameters

int	enum musica

Definition at line 82 of file sound.c.

4.54 sound.c

Go to the documentation of this file.

```
00001
00008 /*****************************************************************************
00009  * INCLUDE HEADER FILES
00010  *****************************************************************************/
00011
00012 #include "../../sound.h"
00013 #include "simpleSDL2audio/audio.h"
00014
00015 /*****************************************************************************
00016  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00017  *****************************************************************************/
00018
00019 #define MUSICA_DIR "../res/sounds/streams"
00020 #define EFECTOS_DIR "../res/sounds/samples"
00021
00022 /*****************************************************************************
00023  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00024  *****************************************************************************/
00025
00026 static char *archivos_musica[] =
00027     {MUSICA_DIR "/main_menu_theme.wav",
00028      MUSICA_DIR "/ranking_theme.wav",
00029      MUSICA_DIR "/credits_theme.wav",
00030      MUSICA_DIR "/playing_theme.wav",
00031      MUSICA_DIR "/pause_menu_theme.wav",
00032      MUSICA_DIR "/game_over.wav"};
00033
00034 static char *archivos_efectos[] =
00035     {EFECTOS_DIR "/click.wav",
00036      EFECTOS_DIR "/jump_original.wav",
00037      EFECTOS_DIR "/crash.wav",
00038      EFECTOS_DIR "/fall_in_water.wav",
00039      EFECTOS_DIR "/low_time_RPI.wav",
00040      EFECTOS_DIR "/goal_reached.wav",
00041      EFECTOS_DIR "/run_completed.wav",
00042      EFECTOS_DIR "/new_max_score.wav",
00043      EFECTOS_DIR "/menu_enter.wav",
00044      EFECTOS_DIR "/saliendo.wav",
00045      EFECTOS_DIR "/no_time.wav"};
00046
00047 static Audio *musica;
00048
00049 static int actual;
00050
00051 /*****************************************************************************
00052  *****************************************************************************
00053                         GLOBAL FUNCTION DEFINITIONS
00054  *****************************************************************************
00055  *****************************************************************************/
00056
00057 bool iniciarSonido(void)
00058 {
00059     if (SDL_Init(SDL_INIT_AUDIO) < 0)
00060     {
00061         return 1;
00062     }
00063
00064     initAudio();
00065     actual = -1;
00066     return true;
00067 }
00068
00069 void destruirSonido(void)
00070 {
00071     endAudio();
00072     freeAudio(musica);
00073     SDL_Quit();
00074 }
00075
00076 void pausarMusica(void)
00077 {
00078     pauseAudio();
00079     actual = -1;
00080 }
00081
00082 void reproducirMusica(int m)
00083 {
00084     if (m != actual)
00085     {
00086         endAudio();
00087         freeAudio(musica);
00088         initAudio();
00089         musica = createAudio(archivos_musica[m], 1, SDL_MIX_MAXVOLUME » 1);
00090         playMusicFromMemory(musica, SDL_MIX_MAXVOLUME » 1);
00091         actual = m;
00092     }
00093 }
```

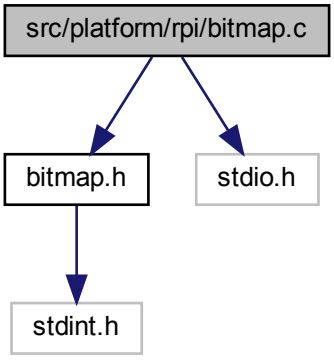
```
00094
00095 void reproducirEfecto(int e)
00096 {
00097     playSound(archivos_efectos[e], SDL_MIX_MAXVOLUME);
00098 }
```

4.55 src/platform/rpi/bitmap.c File Reference

Archivo para manejo de matrices 16x16.

```
#include "bitmap.h"
#include <stdio.h>
```

Include dependency graph for bitmap.c:



Functions

- void printMatriz (matriz_t a)

  Imprime una matriz en consola (para debug)
- void limpiarMatriz (matriz_t a)

  Borra el contenido de una matriz.
- void copiarMatriz (matriz_t destino, const matriz_t desde)

  Copia el contenido de una matriz en otra.
- void matrizAnd (matriz_t a, matriz_t b)

  Dadas dos matrices A y B, se hará la operación "A &= B".
- void matrizOr (matriz_t a, matriz_t b)

  Dadas dos matrices A y B, se hará la operación "A |= B".
- void matrizXor (matriz_t a, matriz_t b)

  Dadas dos matrices A y B, se hará la operación "A ^= B".
- void matrizNot (matriz_t a)

  Dadas una matriz A, se hará la operación "A = ~A".

4.55.1 Detailed Description

Archivo para manejo de matrices 16x16.

Archivo para manejo del display de RPI.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file bitmap.c.

4.55.2 Function Documentation

4.55.2.1 copiarMatriz()

```
void copiarMatriz (
        matriz_t destino,
        const matriz_t desde )
```

Copia el contenido de una matriz en otra.

Parameters

destino	
desde	

Definition at line 36 of file bitmap.c.

4.55.2.2 limpiarMatriz()

```
void limpiarMatriz (
        matriz_t A )
```

Borra el contenido de una matriz.

Parameters

A	

Definition at line 30 of file bitmap.c.

4.55.2.3 matrizAnd() `void matrizAnd (`

```
matriz_t A,
matriz_t B )
```

Dadas dos matrices A y B, se hará la operación "A &= B".

Parameters

A	
B	

Definition at line 42 of file bitmap.c.

4.55.2.4 matrizNot() `void matrizNot (`

```
matriz_t A )
```

Dadas una matriz A, se hará la operación "A = ∼A".

Parameters

A	

Definition at line 60 of file bitmap.c.

4.55.2.5 matrizOr() `void matrizOr (`

```
matriz_t A,
matriz_t B )
```

Dadas dos matrices A y B, se hará la operación "A |= B".

Parameters

A	
B	

Definition at line 48 of file bitmap.c.

4.55.2.6 matrizXor() `void matrizXor (`

```
matriz_t A,
matriz_t B )
```

Dadas dos matrices A y B, se hará la operación "A ^= B".

Parameters

A	
B	

Definition at line 54 of file bitmap.c.

4.55.2.7 printMatriz() void printMatriz (matriz_t A)

Imprime una matriz en consola (para debug)

Parameters

A	

Definition at line 22 of file bitmap.c.

4.56 bitmap.c

Go to the documentation of this file.

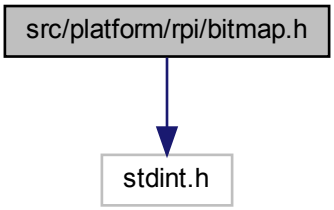
```
00001
00008 /*******************************************************************************
00009  * INCLUDE HEADER FILES
00010  ******************************************************************************/
00011
00012 #include "bitmap.h"
00013
00014 #include <stdio.h>
00015
00016 /*******************************************************************************
00017  *******************************************************************************
00018                         GLOBAL FUNCTION DEFINITIONS
00019  *******************************************************************************
00020  ******************************************************************************/
00021
00022 void printMatriz(matriz_t a)
00023 {
00024     putchar('\n');
00025     for (int i = 0; i < CANT_FILAS; i++, putchar('\n'))
00026         for (int j = 0; j < CANT_FILAS; j++)
00027             putchar((a[i] & (0b1000000000000000 >> j)) ? '#' : '-');
00028 }
00029
00030 void limpiarMatriz(matriz_t a)
00031 {
00032     for (int i = 0; i < CANT_FILAS; i++)
00033         a[i] = 0;
00034 }
00035
00036 void copiarMatriz(matriz_t destino, const matriz_t desde)
00037 {
00038     for (int i = 0; i < CANT_FILAS; i++)
00039         destino[i] = desde[i];
00040 }
00041
00042 void matrizAnd(matriz_t a, matriz_t b)
00043 {
00044     for (int i = 0; i < CANT_FILAS; i++)
00045         a[i] &= b[i];
00046 }
00047
00048 void matrizOr(matriz_t a, matriz_t b)
00049 {
00050     for (int i = 0; i < CANT_FILAS; i++)
```

```
00051         a[i] |= b[i];
00052 }
00053
00054 void matrizXor(matriz_t a, matriz_t b)
00055 {
00056     for (int i = 0; i < CANT_FILAS; i++)
00057         a[i] ^= b[i];
00058 }
00059
00060 void matrizNot(matriz_t a)
00061 {
00062     for (int i = 0; i < CANT_FILAS; i++)
00063         a[i] = ~a[i];
00064 }
```

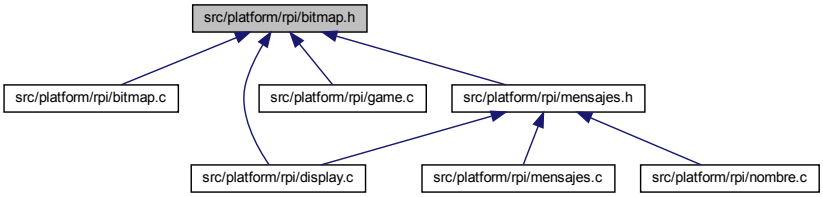
4.57 src/platform/rpi/bitmap.h File Reference

Encabezado del archivo para manejo de matrices 16x16.

`#include <stdint.h>`
Include dependency graph for bitmap.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define CANT_FILAS 16
- #define CANT_COLUMNAS 16

Typedefs

- typedef uint16_t matriz_t[CANT_FILAS]

Functions

- void printMatriz (matriz_t A)

  Imprime una matriz en consola (para debug)
- void limpiarMatriz (matriz_t A)

  Borra el contenido de una matriz.
- void copiarMatriz (matriz_t destino, const matriz_t desde)

  Copia el contenido de una matriz en otra.
- void matrizAnd (matriz_t A, matriz_t B)

  Dadas dos matrices A y B, se hará la operación "A &= B".
- void matrizOr (matriz_t A, matriz_t B)

  Dadas dos matrices A y B, se hará la operación "A |= B".
- void matrizNot (matriz_t A)

  Dadas una matriz A, se hará la operación "A = ∼A".
- void matrizXor (matriz_t A, matriz_t B)

  Dadas dos matrices A y B, se hará la operación "A ^= B".

4.57.1 Detailed Description

Encabezado del archivo para manejo de matrices 16x16.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file bitmap.h.

4.57.2 Macro Definition Documentation

4.57.2.1 CANT_COLUMNAS `#define CANT_COLUMNAS 16`

Definition at line 22 of file bitmap.h.

4.57.2.2 CANT_FILAS `#define CANT_FILAS 16`

Definition at line 21 of file bitmap.h.

4.57.3 Typedef Documentation

4.57.3.1 matriz_t `typedef uint16_t matriz_t[CANT_FILAS]`

Definition at line 28 of file bitmap.h.

4.57.4 Function Documentation

4.57.4.1 copiarMatriz() `void copiarMatriz ( matriz_t destino, const matriz_t desde )`

Copia el contenido de una matriz en otra.

Parameters

destino	
desde	

Definition at line 36 of file bitmap.c.

4.57.4.2 limpiarMatriz() `void limpiarMatriz ( matriz_t A )`

Borra el contenido de una matriz.

Parameters

A	

Definition at line 30 of file bitmap.c.

4.57.4.3 matrizAnd() `void matrizAnd ( matriz_t A, matriz_t B )`

Dadas dos matrices A y B, se hará la operación "A &= B".

Parameters

A	
B	

Definition at line 42 of file bitmap.c.

4.57.4.4 matrizNot() `void matrizNot ( matriz_t A )`

Dadas una matriz A, se hará la operación "A = ∼A".

Parameters

A	

Definition at line 60 of file bitmap.c.

4.57.4.5 matrizOr() `void matrizOr ( matriz_t A, matriz_t B )`

Dadas dos matrices A y B, se hará la operación "A |= B".

Parameters

A	
B	

Definition at line 48 of file bitmap.c.

4.57.4.6 matrizXor() `void matrizXor ( matriz_t A, matriz_t B )`

Dadas dos matrices A y B, se hará la operación "A ^= B".

Parameters

A	
B	

Definition at line 54 of file bitmap.c.

4.57.4.7 printMatriz() `void printMatriz ( matriz_t A )`

Imprime una matriz en consola (para debug)

Parameters

A	

Definition at line 22 of file bitmap.c.

4.58 bitmap.h

Go to the documentation of this file.

```
00001
00008 #ifndef _BITMAP_H_
00009 #define _BITMAP_H_
00010
00011 /*******************************************************************************
00012  * INCLUDE HEADER FILES
00013  ******************************************************************************/
00014
00015 #include <stdint.h>
00016
00017 /*******************************************************************************
00018  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00019  ******************************************************************************/
00020
00021 #define CANT_FILAS 16
00022 #define CANT_COLUMNAS 16
00023
00024 /*******************************************************************************
00025  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00026  ******************************************************************************/
00027
00028 typedef uint16_t matriz_t[CANT_FILAS]; // se define el tipo de dato para trabajar en el display, cada
       elemento del array es una fila
00029
00030 /*******************************************************************************
00031  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00032  ******************************************************************************/
00033
00039 void printMatriz(matriz_t A);
00040
00046 void limpiarMatriz(matriz_t A);
00047
00054 void copiarMatriz(matriz_t destino, const matriz_t desde);
00055
00062 void matrizAnd(matriz_t A, matriz_t B);
00063
00070 void matrizOr(matriz_t A, matriz_t B);
00071
00077 void matrizNot(matriz_t A);
00078
00085 void matrizXor(matriz_t A, matriz_t B);
00086
00087 /*******************************************************************************
00088  ******************************************************************************/
00089
00090 #endif // _BITMAP_H_
```

4.59 disdrv.h

```
00001
00012 #ifndef DISDRV_H
00013 #define DISDRV_H
00014
00015 /*******************************************************************************
00016  * INCLUDE HEADER FILES
00017  ******************************************************************************/
00018
00019 #include <stdint.h>
00020
00021 /*******************************************************************************
00022  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00023  ******************************************************************************/
00024
00025 #define DISP_CANT_X_DOTS 16
00026 #define DISP_CANT_Y_DOTS 16
00027
00028 #define DISP_MIN 0
```

```
00029 #define DISP_MAX_X (DISP_MIN + DISP_CANT_X_DOTS - 1) // = 15
00030 #define DISP_MAX_Y (DISP_MIN + DISP_CANT_Y_DOTS - 1) // = 15
00031
00032 /*******************************************************************************
00033  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00034  ******************************************************************************/
00035
00036 typedef struct
00037 {
00038     uint8_t x; // coordenada x del display
00039     uint8_t y; // coordenada y del display
00040 } dcoord_t;
00041
00042 typedef enum
00043 {
00044     D_OFF,
00045     D_ON
00046 } dlevel_t; // Valores posibles para cada LED
00047
00048 /*******************************************************************************
00049  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00050  ******************************************************************************/
00051
00052 // Display Services
00053
00057 void disp_init(void);
00058
00062 void disp_clear(void);
00063
00071 void disp_write(dcoord_t coord, dlevel_t val);
00072
00076 void disp_update(void);
00077
00078 /*******************************************************************************
00079  ******************************************************************************/
00080
00081 #endif // DISDRV_H
```

4.60 joydrv.h

```
00001
00012 #ifndef JOYDRV_H
00013 #define JOYDRV_H
00014
00015 /*******************************************************************************
00016  * INCLUDE HEADER FILES
00017  ******************************************************************************/
00018
00019 #include <stdint.h>
00020
00021 /*******************************************************************************
00022  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00023  ******************************************************************************/
00024
00025 // Las coordenadas del joystick varían entre -128 y 127 para cada coordenada
00026 #define JOY_MAX_POS 127
00027 #define JOY_MAX_NEG -128
00028
00029 /*******************************************************************************
00030  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00031  ******************************************************************************/
00032
00033 typedef struct
00034 {
00035     int8_t x;
00036     int8_t y;
00037 } jcoord_t;
00038
00039 typedef enum
00040 {
00041     J_NOPRESS,
00042     J_PRESS
00043 } jswitch_t;
00044
00045 /*******************************************************************************
00046  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00047  ******************************************************************************/
00048
00052 void joy_init(void);
00053
00057 int joy_update(void); // void
00058
00063 jcoord_t joy_get_coord(void);
00064
```

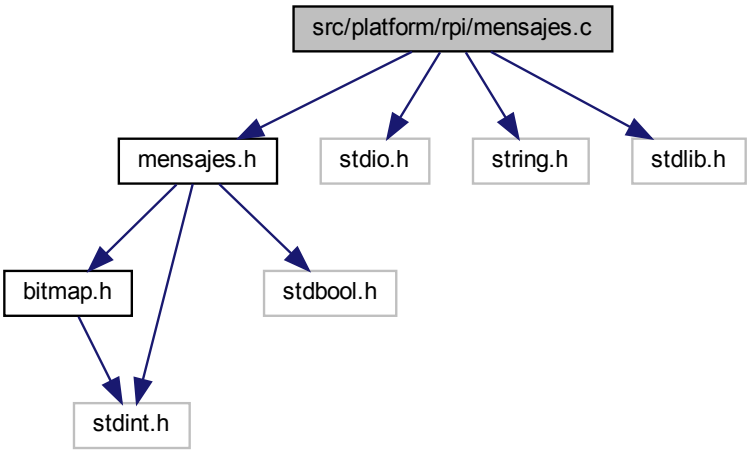
```
00069 jswitch_t joy_get_switch(void);
00070
00071 /*******************************************************************************
00072  ******************************************************************************/
00073
00074 #endif // JOYDRV_H
```

4.61 src/platform/rpi/mensajes.c File Reference

Permite codificar strings en formato renglon para mostrar en display.

```
#include "mensajes.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for mensajes.c:



Macros

- #define INDEX_ESPACIO 26
- #define INDEX_CERO 27
- #define INDEX_FULL 37
- #define CANT_SIMBOLOS 38
- #define PEDIR_FULL -1
- #define ANCHO_MAXIMO 5

Functions

- void printRenglon (renglon_t r)

  imprime dos renglon en consola (para debuggear)
- void borrarRenglon (renglon_t r)

  elimina el contenido del Renglon
- void renglonShiftDer (renglon_t r, uint16_t s)

Desplaza a la derecha el contenido de un Renglon.

- void renglonShiftIzq (renglon_t r, uint16_t s)

  Desplaza a la izquierda el contenido de un Renglon.
- void renglonOr (renglon_t r, renglon_t s)

  Se ejecuta la operación "r |= s".
- void renglonAnd (renglon_t r, renglon_t s)

  Se ejecuta la operación "r &= s".
- void renglonNot (renglon_t r)

  Se invierte el contenido del Renglon (se obtiene el complemento)
- void copiarRenglon (renglon_t r1, const renglon_t r2)

  copia el contenido de r2 en r1
- void copiarMatrizRenglon (matriz_t m, const renglon_t r, int pos)

  copia un Renglon sobre una Matriz a partir de una fila especificada
- bool renglonIzquierdoLibre (mensaje_t *msj)

  indica si la parte izquierda del renglón de un mensaje está vacía
- void charARenglon (char c, renglon_t r)

  Convierte un caracter a Renglon, ubicándolo en el extremo izquierdo del Renglon.
- void charAMatriz (char c, matriz_t m, const int coord[ ])

  Escribe un caracter en una matriz, a partir de las posiciones (x,y) (Extremo superior izdo)
- void uintARenglon (uint16_t n, renglon_t r)

  Convierte un entero no signado a Renglon, ubicándolo en el extremo izquierdo del renglon.
- void reemplazarLetra (renglon_t r, char c, int j)

  (re)escribe sobre el Renglon un caracter dado a partir de la columna j
- mensaje_t mensaje (char *msj, int pos, bool repetir)

  constructor de la variable mensaje_t
- void moverMensaje (mensaje_t *msj)

  desplaza el contenido del Renglon doble hacia la izquierda
- void concatenarLetraMensaje (char c, mensaje_t *msj)

  agrega una letra al string del mensaje y también al renglon
- void reemplazarUltLetraMensaje (char c, mensaje_t *msj)

  reemplaza la última letra en el string del mensaje y también del renglon

4.61.1 Detailed Description

Permite codificar strings en formato renglon para mostrar en display.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file mensajes.c.

4.61.2 Macro Definition Documentation

4.61.2.1 ANCHO_MAXIMO `#define ANCHO_MAXIMO 5`

Definition at line 28 of file mensajes.c.

4.61.2.2 CANT_SIMBOLOS `#define CANT_SIMBOLOS 38`

Definition at line 25 of file mensajes.c.

4.61.2.3 INDEX_CERO `#define INDEX_CERO 27`

Definition at line 23 of file mensajes.c.

4.61.2.4 INDEX_ESPACIO `#define INDEX_ESPACIO 26`

Definition at line 22 of file mensajes.c.

4.61.2.5 INDEX_FULL `#define INDEX_FULL 37`

Definition at line 24 of file mensajes.c.

4.61.2.6 PEDIR_FULL `#define PEDIR_FULL -1`

Definition at line 27 of file mensajes.c.

4.61.3 Function Documentation

4.61.3.1 borrarRenglon() `void borrarRenglon ( renglon_t r )`

elimina el contenido del Renglon

Parameters

r	Renglon a limpiar

Definition at line 120 of file mensajes.c.

4.61.3.2 charAMatriz()

```
void charAMatriz (
        char c,
        matriz_t m,
        const int coord[] )
```

Escribe un caracter en una matriz, a partir de las posiciones (x,y) (Extremo superior izdo)

Parameters

c	caracter
m	Matriz
x	
y	

Definition at line 196 of file mensajes.c.

4.61.3.3 charARenglon()

```
void charARenglon (
        char c,
        renglon_t r )
```

Convierte un caracter a Renglon, ubicándolo en el extremo izquierdo del Renglon.

Parameters

c	caracter
r	Renglon

Definition at line 176 of file mensajes.c.

4.61.3.4 concatenarLetraMensaje()

```
void concatenarLetraMensaje (
        char c,
        mensaje_t * msj )
```

agrega una letra al string del mensaje y también al renglon

Parameters

c	
msj	

Definition at line 314 of file mensajes.c.

4.61.3.5 copiarMatrizRenglon() `void copiarMatrizRenglon (`

```
matriz_t m,
const renglon_t r,
int pos )
```

copia un Renglon sobre una Matriz a partir de una fila especificada

Parameters

m	Matriz destino
r	Renglon origen
pos	Fila de inicio

Definition at line 162 of file mensajes.c.

4.61.3.6 copiarRenglon() `void copiarRenglon (`

```
renglon_t r1,
const renglon_t r2 )
```

copia el contenido de r2 en r1

Parameters

r1	Renglon destino
r2	Renglon origen

Definition at line 156 of file mensajes.c.

4.61.3.7 mensaje() `mensaje_t mensaje (`

```
char * msj,
int pos,
bool repetir )
```

constructor de la variable mensaje_t

Parameters

msj	String que se desea convertir a mensaje
pos	fila sobre la que se deberá mostrar en display
repetir	si se repetirá una vez terminado de mostrar

Returns

mensaje_t

Definition at line 242 of file mensajes.c.

4.61.3.8 moverMensaje() `void moverMensaje (`

```
mensaje_t * msj )
```

desplaza el contenido del Renglon doble hacia la izquierda

Parameters

msj	puntero a la variable mensaje_t

Definition at line 283 of file mensajes.c.

4.61.3.9 printRenglon() `void printRenglon (`

```
renglon_t r )
```

imprime dos renglon en consola (para debuggear)

Parameters

r	renglon a imprimir

Definition at line 106 of file mensajes.c.

4.61.3.10 reemplazarLetra() `void reemplazarLetra (`

```
renglon_t r,
char c,
int j )
```

(re)escribe sobre el Renglon un caracter dado a partir de la columna j

Parameters

r	Renglon
c	caracter
j	columna sobre la que se quiere escribir

Definition at line 230 of file mensajes.c.

4.61.3.11 reemplazarUltLetraMensaje() `void reemplazarUltLetraMensaje (`

```
char c,
mensaje_t * msj )
```

reemplaza la última letra en el string del mensaje y también del renglon

Parameters

c	
msj	

Definition at line 337 of file mensajes.c.

4.61.3.12 renglonAnd() `void renglonAnd (`

```
renglon_t r,
renglon_t s )
```

Se ejecuta la operación "r &= s".

Parameters

r	primer operando AND
s	segundo operando AND

Definition at line 144 of file mensajes.c.

4.61.3.13 renglonIzquierdoLibre() `bool renglonIzquierdoLibre (`

```
mensaje_t * msj )
```

indica si la parte izquierda del renglón de un mensaje está vacía

Parameters

r	Renglon a chequear

Returns

true

false

Definition at line 168 of file mensajes.c.

4.61.3.14 renglonNot() `void renglonNot (`

```
renglon_t r )
```

Se invierte el contenido del Renglon (se obtiene el complemento)

Parameters

r	Renglon a invertir

Definition at line 150 of file mensajes.c.

4.61.3.15 renglonOr()

```
void renglonOr (
    renglon_t r,
    renglon_t s )
```

Se ejecuta la operación "r |= s".

Parameters

r	primer operando OR
s	segundo operando OR

Definition at line 138 of file mensajes.c.

4.61.3.16 renglonShiftDer()

```
void renglonShiftDer (
    renglon_t r,
    uint16_t s )
```

Desplaza a la derecha el contenido de un Renglon.

Parameters

r	Renglon a desplazar
s	cantidad de espacios

Definition at line 126 of file mensajes.c.

4.61.3.17 renglonShiftIzq()

```
void renglonShiftIzq (
    renglon_t r,
    uint16_t s )
```

Desplaza a la izquierda el contenido de un Renglon.

Parameters

r	Renglon a desplazar
s	cantidad de espacios

Definition at line 132 of file mensajes.c.

4.61.3.18 uintARenglon() `void uintARenglon (`

```
uint16_t n,
renglon_t r )
```

Convierte un entero no signado a Renglon, ubicándolo en el extremo izquierdo del renglon.

Parameters

n	entero no signado de 16 bits
r	Renglon

Definition at line 205 of file mensajes.c.

4.62 mensajes.c

Go to the documentation of this file.

```
00001
00008 /*******************************************************************************
00009  * INCLUDE HEADER FILES
00010  ******************************************************************************/
00011
00012 #include "mensajes.h"
00013
00014 #include <stdio.h>
00015 #include <string.h>
00016 #include <stdlib.h>
00017
00018 /*******************************************************************************
00019  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00020  ******************************************************************************/
00021
00022 #define INDEX_ESPACIO 26
00023 #define INDEX_CERO 27
00024 #define INDEX_FULL 37
00025 #define CANT_SIMBOLOS 38
00026
00027 #define PEDIR_FULL -1
00028 #define ANCHO_MAXIMO 5
00029
00030 /*******************************************************************************
00031  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00032  ******************************************************************************/
00033
00034 /*******************************************************************************
00035  * VARIABLES WITH GLOBAL SCOPE
00036  ******************************************************************************/
00037
00038 // +ej: unsigned int anio_actual;+
00039
00040 /*******************************************************************************
00041  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00042  ******************************************************************************/
00043
00050 static int getLongitud(char a);
00051
00052 /*******************************************************************************
00053  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00054  ******************************************************************************/
00055
00056 static const int longitudes[] = {3, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 5, 4, 3, 3, 3, 3, 3, 3, 3, 5,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5}; // sin contar Ñ (+ espacio + numeros + FULL)
00057
00058 /*******************************************************************************
00059  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00060  ******************************************************************************/
00061
```

```
00062 static uint16_t char_index[][TAM_RENGLON] = {{0x4000, 0xA000, 0xE000, 0xA000, 0xA000}, // A
00063                                             {0xC000, 0xA000, 0xC000, 0xA000, 0xC000},
00064                                             {0x4000, 0xA000, 0x8000, 0xA000, 0x4000},
00065                                             {0xC000, 0xA000, 0xA000, 0xA000, 0xC000},
00066                                             {0xE000, 0x8000, 0xC000, 0x8000, 0xE000},
00067                                             {0xE000, 0x8000, 0xC000, 0x8000, 0x8000},
00068                                             {0x6000, 0x8000, 0xA000, 0xA000, 0x6000},
00069                                             {0xA000, 0xA000, 0xE000, 0xA000, 0xA000},
00070                                             {0x8000, 0x8000, 0x8000, 0x8000, 0x8000},
00071                                             {0x2000, 0x2000, 0xA000, 0xA000, 0x4000},
00072                                             {0xA000, 0xA000, 0xC000, 0xA000, 0xA000},
00073                                             {0x8000, 0x8000, 0x8000, 0x8000, 0xE000},
00074                                             {0x8800, 0xD800, 0xA800, 0xA800, 0x8800},
00075                                             {0x9000, 0xD000, 0xB000, 0x9000, 0x9000},
00076                                             {0x4000, 0xA000, 0xA000, 0xA000, 0x4000},
00077                                             {0xC000, 0xA000, 0xA000, 0xC000, 0x8000},
00078                                             {0x4000, 0xA000, 0xA000, 0xA000, 0x6000},
00079                                             {0xC000, 0xA000, 0xC000, 0xA000, 0xA000},
00080                                             {0x6000, 0x8000, 0x4000, 0x2000, 0xC000},
00081                                             {0xE000, 0x4000, 0x4000, 0x4000, 0x4000},
00082                                             {0xA000, 0xA000, 0xA000, 0xA000, 0x4000},
00083                                             {0xA000, 0xA000, 0xA000, 0x4000, 0x4000},
00084                                             {0x8800, 0xA800, 0xA800, 0x5000, 0x5000},
00085                                             {0xA000, 0xA000, 0x4000, 0xA000, 0xA000},
00086                                             {0xA000, 0xA000, 0x4000, 0x4000, 0x4000},
00087                                             {0xE000, 0x2000, 0x4000, 0x8000, 0xE000}, // Z
00088                                             {0x0000, 0x0000, 0x0000, 0x0000, 0x0000}, // espacio
00089                                             {0x4000, 0xA000, 0xA000, 0xA000, 0x4000}, // 0
00090                                             {0x4000, 0xC000, 0x4000, 0x4000, 0xE000},
00091                                             {0x4000, 0xA000, 0x2000, 0x4000, 0xE000},
00092                                             {0xC000, 0x2000, 0x4000, 0x2000, 0xC000},
00093                                             {0xA000, 0xA000, 0xE000, 0x2000, 0x2000},
00094                                             {0xE000, 0x8000, 0xC000, 0x2000, 0xC000},
00095                                             {0x4000, 0x8000, 0xC000, 0xA000, 0xC000},
00096                                             {0xE000, 0x2000, 0x2000, 0x4000, 0x4000},
00097                                             {0x4000, 0xA000, 0x4000, 0xA000, 0x4000},
00098                                             {0x4000, 0xA000, 0x6000, 0x2000, 0x4000},  // 9
00099                                             {0xF800, 0xF800, 0xF800, 0xF800, 0xF800}}; // TODO (FULL)
00100
00101 /*******************************************************************************
00102  *******************************************************************************
00103                         GLOBAL FUNCTION DEFINITIONS
00104  *******************************************************************************
00105  *******************************************************************************/
00106 void printRenglon(renglon_t r)
00107 {
00108     putchar('\n');
00109     for (int i = 0; i < TAM_RENGLON; i++, putchar('\n'))
00110     {
00111         for (int j = 0; j < 2 * CANT_FILAS; j++)
00112         {
00113             putchar((r[i].completo & (0x80000000 >> j)) ? '#' : '-');
00114             if (j == CANT_FILAS - 1)
00115                 putchar('|');
00116         }
00117     }
00118 }
00119
00120 void borrarRenglon(renglon_t r)
00121 {
00122     for (int i = 0; i < TAM_RENGLON; i++)
00123         r[i].completo = 0;
00124 }
00125
00126 void renglonShiftDer(renglon_t r, uint16_t s)
00127 {
00128     for (int i = 0; i < TAM_RENGLON; i++)
00129         r[i].completo >>= s;
00130 }
00131
00132 void renglonShiftIzq(renglon_t r, uint16_t s)
00133 {
00134     for (int i = 0; i < TAM_RENGLON; i++)
00135         r[i].completo <<= s;
00136 }
00137
00138 void renglonOr(renglon_t r, renglon_t s)
00139 {
00140     for (int i = 0; i < TAM_RENGLON; i++)
00141         r[i].completo |= s[i].completo;
00142 }
00143
00144 void renglonAnd(renglon_t r, renglon_t s)
00145 {
00146     for (int i = 0; i < TAM_RENGLON; i++)
00147         r[i].completo &= s[i].completo;
00148 }
```

```
00149
00150 void renglonNot(renglon_t r)
00151 {
00152     for (int i = 0; i < TAM_RENGLON; i++)
00153         r[i].completo = ~r[i].completo;
00154 }
00155
00156 void copiarRenglon(renglon_t r1, const renglon_t r2)
00157 {
00158     for (int i = 0; i < TAM_RENGLON; i++)
00159         r1[i].completo = r2[i].completo;
00160 }
00161
00162 void copiarMatrizRenglon(matriz_t m, const renglon_t r, int pos)
00163 {
00164     for (int i = 0; i < TAM_RENGLON; i++)
00165         m[pos + i] = r[i].mitad_izq;
00166 }
00167
00168 bool renglonIzquierdoLibre(mensaje_t *msj)
00169 {
00170     for (int i = 0; i < TAM_RENGLON; i++)
00171         if ((msj->renglon)[i].mitad_izq)
00172             return false;
00173     return true;
00174 }
00175
00176 void charARenglon(char c, renglon_t r)
00177 {
00178     int indice;
00179
00180     if (c == ' ' || !c)
00181         indice = INDEX_ESPACIO;
00182     else if ('0' <= c && c <= '9')
00183         indice = INDEX_CERO + c - '0';
00184     else if ('A' <= c && c <= 'Z')
00185         indice = c - 'A';
00186     else
00187         indice = INDEX_FULL;
00188
00189     for (int i = 0; i < TAM_RENGLON; i++)
00190     {
00191         r[i].mitad_izq = char_index[indice][i];
00192         r[i].mitad_der = 0;
00193     }
00194 }
00195
00196 void charAMatriz(char c, matriz_t m, const int coord[])
00197 {
00198     limpiarMatriz(m);
00199     renglon_t r;
00200     charARenglon(c, r);
00201     renglonShiftDer(r, coord[0]);
00202     copiarMatrizRenglon(m, r, coord[1]);
00203 }
00204
00205 void uintARenglon(uint16_t n, renglon_t r)
00206 {
00207     renglon_t renglon_aux;
00208     int j = 0, resto = CANT_COLUMNAS;
00209     uint16_t div = 10000;
00210
00211     while (n % div == n)
00212         div /= 10;
00213
00214     while (n)
00215     {
00216         uint16_t aux = n % div;
00217         n = (n - aux) / div;
00218         resto = CANT_COLUMNAS - j - longitudes[INDEX_CERO + n];
00219         if (resto < 0)
00220             break;
00221         charARenglon(n + '0', renglon_aux);
00222         renglonShiftDer(renglon_aux, j);
00223         renglonOr(r, renglon_aux);
00224         j += longitudes[INDEX_CERO + n] + 1;
00225         n = aux;
00226         div /= 10;
00227     }
00228 }
00229
00230 void reemplazarLetra(renglon_t r, char c, int j)
00231 {
00232     renglon_t full, letra;
00233     charARenglon(c, letra);
00234     renglonShiftDer(letra, j);
00235     charARenglon(PEDIR_FULL, full);
```

```
00236     renglonShiftDer(full, j);
00237     renglonNot(full);
00238     renglonAnd(r, full);
00239     renglonOr(r, letra);
00240 }
00241
00242 mensaje_t mensaje(char *msj, int pos, bool repetir)
00243 {
00244     mensaje_t mensaje = {.posicion = pos, .habilitacion = true, .repetir_msj = repetir};
00245     borrarRenglon(mensaje.renglon);
00246     strcpy(mensaje.msj, msj);
00247     int longitud_parcial = strlen(mensaje.msj);
00248
00249     int j = 0; // a partir de donde voy a escribir la proxima vez
00250     int i;
00251     for (i = 0; i < longitud_parcial; i++)
00252     {                   // rellena el mensaje por primera vez
00253         char c = msj[i]; // el caracter que debo escribir
00254
00255         renglon_t letra;
00256         charARenglon(c, letra); // letra contiene la letra provisoria pasada a renglón
00257
00258         int ancho = getLongitud(c);
00259         if (2 * CANT_COLUMNAS - j - ancho < 0)
00260             break; // lo que me quedaría libre si escribo
00261
00262         renglonShiftDer(letra, j);          // muevo la letra sobre renglon
00263         renglonOr(mensaje.renglon, letra); // escribo en el renglon
00264         j += ancho + 1;                    // dejo un espacio entre letra y letra
00265     }
00266
00267     mensaje.index = i;
00268     mensaje.j = j;
00269
00270     mensaje.mover_texto = (i < longitud_parcial) || (j > CANT_COLUMNAS + 1); // veo si entra el
      mensaje en el renglon izquierdo
00271
00272     if (mensaje.mover_texto)
00273     {
00274         strcat(mensaje.msj, "     ");
00275         mensaje.longitud = strlen(mensaje.msj);
00276     }
00277     else
00278         mensaje.longitud = longitud_parcial;
00279
00280     return mensaje;
00281 }
00282
00283 void moverMensaje(mensaje_t *msj)
00284 {
00285     if (!(msj->mover_texto) || !(msj->habilitacion)) // tengo permitido mover el mensaje?
00286         return;
00287
00288     renglonShiftIzq(msj->renglon, 1);
00289     msj->j--;
00290
00291     if (!(msj->repetir_msj) && msj->index >= msj->longitud) // tengo que repetirlo cuando termine de
      recorrerlo?
00292     {
00293         if (renglonIzquierdoLibre(msj)) // termine de pasar el mensaje?
00294             msj->habilitacion = false;
00295         return;
00296     }
00297
00298     char c = (msj->msj)[msj->index];
00299     int ancho = getLongitud(c);
00300
00301     if (2 * CANT_COLUMNAS - msj->j - ancho < 0) // me queda espacio para poner la siguiente letra?
00302         return;
00303
00304     // pongo la siguiente letra
00305     renglon_t letra;
00306     charARenglon(c, letra);
00307     renglonShiftDer(letra, msj->j);
00308     renglonOr(msj->renglon, letra);
00309     msj->j += ancho + 1; // dejo un espacio entre letra y letra
00310     if (++msj->index >= msj->longitud && msj->repetir_msj)
00311         msj->index = 0;
00312 }
00313
00314 void concatenarLetraMensaje(char c, mensaje_t *msj)
00315 {
00316     if (msj->index == L_MAX - 1)
00317         return;
00318
00319     (msj->msj)[msj->index++] = c;
00320     (msj->msj)[msj->index] = 'A';
```

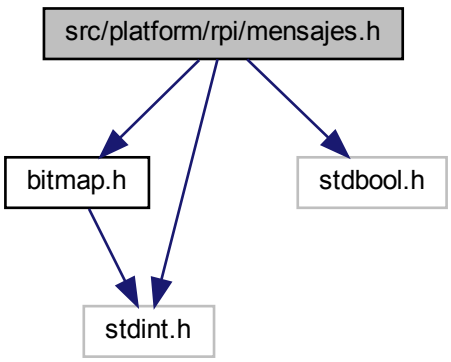
```
00321     (msj->msj)[msj->index + 1] = '\0';
00322     msj->longitud++;
00323
00324     msj->j += getLongitud(c) + 1;
00325     while (CANT_COLUMNAS - msj->j - ANCHO_MAXIMO < 0) // corro a la izquierda hasta asegurarme que va
      a entrar cualquier letra
00326     {
00327         renglonShiftIzq(msj->renglon, 1);
00328         msj->j--;
00329     }
00330
00331     renglon_t letra;
00332     charARenglon('A', letra);
00333     renglonShiftDer(letra, msj->j);
00334     renglonOr(msj->renglon, letra);
00335 }
00336
00337 void reemplazarUltLetraMensaje(char c, mensaje_t *msj)
00338 {
00339     (msj->msj)[msj->index] = c;
00340     reemplazarLetra(msj->renglon, c, msj->j);
00341 }
00342
00343 /*******************************************************************************
00344  *******************************************************************************
00345             LOCAL FUNCTION DEFINITIONS
00346  *******************************************************************************
00347  *******************************************************************************/
00348
00349 static int getLongitud(char c)
00350 {
00351     if (c == ' ' || !c)
00352         return longitudes[INDEX_ESPACIO];
00353     else if ('0' <= c && c <= '9')
00354         return longitudes[INDEX_CERO + c - '0'];
00355     else if ('A' <= c && c <= 'Z')
00356         return longitudes[c - 'A'];
00357     else
00358         return longitudes[INDEX_FULL];
00359 }
```

4.63 src/platform/rpi/mensajes.h File Reference

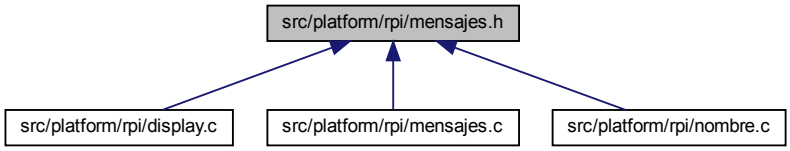
Encabezado de mensajes, con definiciones sobre tipos de datos y funciones.

```
#include "bitmap.h"
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for mensajes.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- union renglon_t
- struct Mensaje

Macros

- #define TAM_RENGLON 5
- #define POS_MSJ1 2
- #define POS_MSJ2 9
- #define POS_MSJ3 5
- #define L_MAX 64

Typedefs

- typedef struct Mensaje **mensaje_t**

Functions

- void printRenglon (renglon_t r)

  imprime dos renglon en consola (para debuggear)
- void borrarRenglon (renglon_t r)

  elimina el contenido del Renglon
- void renglonShiftDer (renglon_t r, uint16_t s)

  Desplaza a la derecha el contenido de un Renglon.
- void renglonShiftIzq (renglon_t r, uint16_t s)

  Desplaza a la izquierda el contenido de un Renglon.
- void renglonOr (renglon_t r, renglon_t s)

  Se ejecuta la operación "r |= s".
- void renglonAnd (renglon_t r, renglon_t s)

  Se ejecuta la operación "r &= s".
- void renglonNot (renglon_t r)

  Se invierte el contenido del Renglon (se obtiene el complemento)
- void copiarRenglon (renglon_t r1, const renglon_t r2)

  copia el contenido de r2 en r1
- void copiarMatrizRenglon (matriz_t m, const renglon_t r, int pos)

  copia un Renglon sobre una Matriz a partir de una fila especificada

- bool renglonIzquierdoLibre (mensaje_t *msj)

  indica si la parte izquierda del renglón de un mensaje está vacía
- void charARenglon (char c, renglon_t r)

  Convierte un caracter a Renglon, ubicándolo en el extremo izquierdo del Renglon.
- void charAMatriz (char c, matriz_t m, const int coord[ ])

  Escribe un caracter en una matriz, a partir de las posiciones (x,y) (Extremo superior izdo)
- void uintARenglon (uint16_t n, renglon_t r)

  Convierte un entero no signado a Renglon, ubicándolo en el extremo izquierdo del renglon.
- void reemplazarLetra (renglon_t r, char c, int j)

  (re)escribe sobre el Renglon un caracter dado a partir de la columna j
- mensaje_t mensaje (char *msj, int pos, bool repetir)

  constructor de la variable mensaje_t
- void moverMensaje (mensaje_t *msj)

  desplaza el contenido del Renglon doble hacia la izquierda
- void concatenarLetraMensaje (char c, mensaje_t *msj)

  agrega una letra al string del mensaje y también al renglon
- void reemplazarUltLetraMensaje (char c, mensaje_t *msj)

  reemplaza la última letra en el string del mensaje y también del renglon

4.63.1 Detailed Description

Encabezado de mensajes, con definiciones sobre tipos de datos y funciones.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file mensajes.h.

4.63.2 Macro Definition Documentation

4.63.2.1 L_MAX `#define L_MAX 64`

Definition at line 29 of file mensajes.h.

4.63.2.2 POS_MSJ1 `#define POS_MSJ1 2`

Definition at line 26 of file mensajes.h.

4.63.2.3 POS_MSJ2 `#define POS_MSJ2 9`

Definition at line 27 of file mensajes.h.

4.63.2.4 POS_MSJ3 `#define POS_MSJ3 5`

Definition at line 28 of file mensajes.h.

4.63.2.5 TAM_RENGLON `#define TAM_RENGLON 5`

Definition at line 24 of file mensajes.h.

4.63.3 Function Documentation

4.63.3.1 borrarRenglon() `void borrarRenglon ( renglon_t r )`

elimina el contenido del Renglon

Parameters

r	Renglon a limpiar

Definition at line 120 of file mensajes.c.

4.63.3.2 charAMatriz() `void charAMatriz ( char c, matriz_t m, const int coord[] )`

Escribe un caracter en una matriz, a partir de las posiciones (x,y) (Extremo superior izdo)

Parameters

c	caracter
m	Matriz
x	
y	

Definition at line 196 of file mensajes.c.

4.63.3.3 charARenglon() `void charARenglon (`

```
char c,
renglon_t r )
```

Convierte un caracter a Renglon, ubicándolo en el extremo izquierdo del Renglon.

Parameters

c	caracter
r	Renglon

Definition at line 176 of file mensajes.c.

4.63.3.4 concatenarLetraMensaje() `void concatenarLetraMensaje (`

```
char c,
mensaje_t * msj )
```

agrega una letra al string del mensaje y también al renglon

Parameters

c	
msj	

Definition at line 314 of file mensajes.c.

4.63.3.5 copiarMatrizRenglon() `void copiarMatrizRenglon (`

```
matriz_t m,
const renglon_t r,
int pos )
```

copia un Renglon sobre una Matriz a partir de una fila especificada

Parameters

m	Matriz destino
r	Renglon origen
pos	Fila de inicio

Definition at line 162 of file mensajes.c.

4.63.3.6 copiarRenglon() `void copiarRenglon (`

```
renglon_t r1,
const renglon_t r2 )
```

copia el contenido de r2 en r1

Parameters

r1	Renglon destino
r2	Renglon origen

Definition at line 156 of file mensajes.c.

4.63.3.7 mensaje() `mensaje_t mensaje (`

```
char * msj,
int pos,
bool repetir )
```

constructor de la variable mensaje_t

Parameters

msj	String que se desea convertir a mensaje
pos	fila sobre la que se deberá mostrar en display
repetir	si se repetirá una vez terminado de mostrar

Returns

mensaje_t

Definition at line 242 of file mensajes.c.

4.63.3.8 moverMensaje() `void moverMensaje (`

```
mensaje_t * msj )
```

desplaza el contenido del Renglon doble hacia la izquierda

Parameters

msj	puntero a la variable mensaje_t

Definition at line 283 of file mensajes.c.

4.63.3.9 printRenglon() `void printRenglon (`

```
renglon_t r )
```

imprime dos renglon en consola (para debuggear)

Parameters

r	renglon a imprimir

Definition at line 106 of file mensajes.c.

4.63.3.10 reemplazarLetra() `void reemplazarLetra (`

```
renglon_t r,
char c,
int j )
```

(re)escribe sobre el Renglon un caracter dado a partir de la columna j

Parameters

r	Renglon
c	caracter
j	columna sobre la que se quiere escribir

Definition at line 230 of file mensajes.c.

4.63.3.11 reemplazarUltLetraMensaje() `void reemplazarUltLetraMensaje (`

```
char c,
mensaje_t * msj )
```

reemplaza la última letra en el string del mensaje y también del renglon

Parameters

c	
msj	

Definition at line 337 of file mensajes.c.

4.63.3.12 renglonAnd() `void renglonAnd (`

```
renglon_t r,
renglon_t s )
```

Se ejecuta la operación "r &= s".

Parameters

r	primer operando AND
s	segundo operando AND

Definition at line 144 of file mensajes.c.

4.63.3.13 renglonIzquierdoLibre() `bool renglonIzquierdoLibre (`

```
mensaje_t * msj )
```

indica si la parte izquierda del renglón de un mensaje está vacía

Parameters

r	Renglon a chequear

Returns

true

false

Definition at line 168 of file mensajes.c.

4.63.3.14 renglonNot() `void renglonNot (`

```
renglon_t r )
```

Se invierte el contenido del Renglon (se obtiene el complemento)

Parameters

r	Renglon a invertir

Definition at line 150 of file mensajes.c.

4.63.3.15 renglonOr() `void renglonOr (`

```
renglon_t r,
renglon_t s )
```

Se ejecuta la operación "r |= s".

Parameters

r	primer operando OR
s	segundo operando OR

Definition at line 138 of file mensajes.c.

4.63.3.16 renglonShiftDer()

```
void renglonShiftDer (
    renglon_t r,
    uint16_t s )
```

Desplaza a la derecha el contenido de un Renglon.

Parameters

r	Renglon a desplazar
s	cantidad de espacios

Definition at line 126 of file mensajes.c.

4.63.3.17 renglonShiftIzq()

```
void renglonShiftIzq (
    renglon_t r,
    uint16_t s )
```

Desplaza a la izquierda el contenido de un Renglon.

Parameters

r	Renglon a desplazar
s	cantidad de espacios

Definition at line 132 of file mensajes.c.

4.63.3.18 uintARenglon()

```
void uintARenglon (
    uint16_t n,
    renglon_t r )
```

Convierte un entero no signado a Renglon, ubicándolo en el extremo izquierdo del renglon.

Parameters

n	entero no signado de 16 bits
r	Renglon

Definition at line 205 of file mensajes.c.

4.64 mensajes.h

Go to the documentation of this file.

```
00001
00008 #ifndef _MENSAJES_H_
00009 #define _MENSAJES_H_
00010
00011 /*****************************************************************************
00012  * INCLUDE HEADER FILES
00013  *****************************************************************************/
00014
00015 #include "bitmap.h"
00016
00017 #include <stdint.h>
00018 #include <stdbool.h>
00019
00020 /*****************************************************************************
00021  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00022  *****************************************************************************/
00023
00024 #define TAM_RENGLON 5
00025
00026 #define POS_MSJ1 2
00027 #define POS_MSJ2 9
00028 #define POS_MSJ3 5
00029 #define L_MAX 64
00030
00031 /*****************************************************************************
00032  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00033  *****************************************************************************/
00034 typedef union
00035 {
00036     uint32_t completo;
00037     struct
00038     {
00039         uint16_t mitad_der;
00040         uint16_t mitad_izq;
00041     };
00042 } renglon_t[5];
00043
00044 typedef struct Mensaje
00045 {
00046     char msj[L_MAX];
00047     int posicion;
00048     int index;
00049     int longitud;
00050     int j;
00051     bool habilitacion;
00052     bool mover_texto;
00053     bool repetir_msj;
00054     renglon_t renglon;
00055 } mensaje_t;
00056
00057 /*****************************************************************************
00058  * VARIABLE PROTOTYPES WITH GLOBAL SCOPE
00059  *****************************************************************************/
00060
00061 // +ej: extern unsigned int anio_actual;+
00062
00063 /*****************************************************************************
00064  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00065  *****************************************************************************/
00066
00072 void printRenglon(renglon_t r);
00073
00079 void borrarRenglon(renglon_t r);
00080
00087 void renglonShiftDer(renglon_t r, uint16_t s);
00088
00095 void renglonShiftIzq(renglon_t r, uint16_t s);
00096
00103 void renglonOr(renglon_t r, renglon_t s);
00104
00111 void renglonAnd(renglon_t r, renglon_t s);
00112
00118 void renglonNot(renglon_t r);
00119
00126 void copiarRenglon(renglon_t r1, const renglon_t r2);
00127
00135 void copiarMatrizRenglon(matriz_t m, const renglon_t r, int pos);
00136
00144 bool renglonIzquierdoLibre(mensaje_t *msj);
00145
00152 void charARenglon(char c, renglon_t r);
00153
00162 void charAMatriz(char c, matriz_t m, const int coord[]);
```

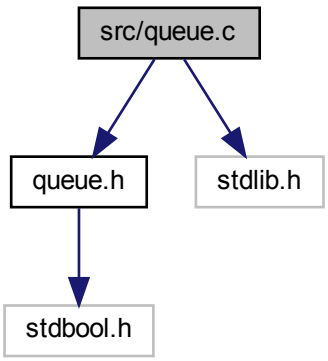
```
00163
00170 void uintARenglon(uint16_t n, renglon_t r); // copia un número a renglon hasta que se acabe el número
       o el renglon (lo 1 q ocurra)
00171
00179 void reemplazarLetra(renglon_t r, char c, int j);
00180
00189 mensaje_t mensaje(char *msj, int pos, bool repetir);
00190
00196 void moverMensaje(mensaje_t *msj);
00197
00204 void concatenarLetraMensaje(char c, mensaje_t *msj);
00205
00212 void reemplazarUltLetraMensaje(char c, mensaje_t *msj);
00213
00214 /*****************************************************************************
00215  *****************************************************************************/
00216
00217 #endif // _MENSAJES_H_
```

4.65 src/queue.c File Reference

Source del modulo queue. Funciones para el manejo de la cola de eventos.

```
#include "queue.h"
#include <stdlib.h>
```

Include dependency graph for queue.c:



Data Structures

- struct nodeT

Typedefs

- typedef struct nodeT **node_t**

Functions

- void queueInsertar (event_t nuevo)

  Agrega un evento a la cola.
- bool queueVacia (void)

  Chequea si la cola está vacía.
- event_t queueSiguienteEvento (void)

  Devuelve el siguiente evento de la cola.
- void destruirQueue (void)

  Destruye la cola de eventos.

4.65.1 Detailed Description

Source del modulo queue. Funciones para el manejo de la cola de eventos.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file queue.c.

4.65.2 Function Documentation

4.65.2.1 destruirQueue() `void destruirQueue ( void )`

Destruye la cola de eventos.

Definition at line 96 of file queue.c.

4.65.2.2 queueInsertar() `void queueInsertar ( event_t nuevo )`

Agrega un evento a la cola.

Definition at line 53 of file queue.c.

4.65.2.3 queueSiguienteEvento() `event_t queueSiguienteEvento ( void )`

Devuelve el siguiente evento de la cola.

Returns

event_t

Definition at line 82 of file queue.c.

4.65.2.4 queueVacia() `bool queueVacia ( void )`

Chequea si la cola está vacía.

Returns

true Vacía

false No vacía

Definition at line 77 of file queue.c.

4.66 queue.c

Go to the documentation of this file.

```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "queue.h"
00017
00018 #include <stdlib.h>
00019
00020 /*******************************************************************************
00021  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00022  ******************************************************************************/
00023
00024 // Estructura del nodo
00025 typedef struct nodeT
00026 {
00027     event_t data;
00028     struct nodeT *next;
00029 } node_t;
00030
00031 /*******************************************************************************
00032  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00033  ******************************************************************************/
00034
00039 static void borrarElemento(void);
00040
00041 /*******************************************************************************
00042  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00043  ******************************************************************************/
00044
00045 static node_t *front = NULL, *back = NULL;
00046
00047 /*******************************************************************************
00048  *******************************************************************************
00049                         GLOBAL FUNCTION DEFINITIONS
00050  *******************************************************************************
00051  ******************************************************************************/
00052
```

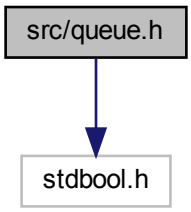
```
00053 void queueInsertar(event_t nuevo)
00054 {
00055     node_t *temp = (node_t *)malloc(sizeof(node_t));
00056 
00057     temp->data = nuevo;
00058 
00059     if (front == NULL)
00060     {
00061         front = temp;
00062         front->next = NULL;
00063     }
00064     else if (back == NULL)
00065     {
00066         back = temp;
00067         front->next = back;
00068         back->next = NULL;
00069     }
00070     else
00071     {
00072         back->next = temp;
00073         back = back->next;
00074     }
00075 }
00076 
00077 bool queueVacia(void)
00078 {
00079     return front == NULL;
00080 }
00081 
00082 event_t queueSiguienteEvento(void)
00083 {
00084     if (front == NULL)
00085     {
00086         return NADA;
00087     }
00088     else
00089     {
00090         event_t r = front->data;
00091         borrarElemento();
00092         return r;
00093     }
00094 }
00095 
00096 void destruirQueue(void)
00097 {
00098     while (front != NULL)
00099     {
00100         queueSiguienteEvento();
00101     }
00102 }
00103 
00104 /********************************************************************************
00105  ********************************************************************************
00106                         LOCAL FUNCTION DEFINITIONS
00107  ********************************************************************************
00108  ********************************************************************************/
00109 
00110 static void borrarElemento(void)
00111 {
00112     if (front != NULL)
00113     {
00114         node_t *temp = front;
00115         front = front->next;
00116         free(temp);
00117         if (front == NULL)
00118         {
00119             back = NULL;
00120         }
00121     }
00122 }
```

4.67 src/queue.h File Reference

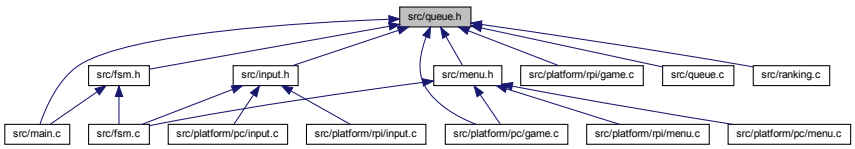
Header del modulo queue. Prototipos de funciones de interaccion con la cola de eventos.

```
#include <stdbool.h>
```

Include dependency graph for queue.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef int event_t

Enumerations

- enum **eventos** { **NADA** = -1 , **SALIR** = 0 , **GAME_OVER** , **FORCE_SALIR** }
- enum **eventos_tecla** {
  NO_MOVER = -1 , **ESC** = 59 , **BORRAR** = 63 , **ENTER** = 67 ,
  IZDA = 82 , **DCHA** , **ARRIBA** , **ABAJO** }

Functions

- void queueInsertar (event_t)

  Agrega un evento a la cola.
- bool queueVacia (void)

  Chequea si la cola está vacía.
- event_t queueSiguienteEvento (void)

  Devuelve el siguiente evento de la cola.
- void destruirQueue (void)

  Destruye la cola de eventos.

4.67.1 Detailed Description

Header del modulo queue. Prototipos de funciones de interaccion con la cola de eventos.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file queue.h.

4.67.2 Typedef Documentation

4.67.2.1 event_t `typedef int event_t`

Definition at line 28 of file queue.h.

4.67.3 Enumeration Type Documentation

4.67.3.1 eventos `enum eventos`

Definition at line 31 of file queue.h.

4.67.3.2 eventos_tecla `enum eventos_tecla`

Definition at line 40 of file queue.h.

4.67.4 Function Documentation

4.67.4.1 destruirQueue() `void destruirQueue ( void )`

Destruye la cola de eventos.

Definition at line 96 of file queue.c.

4.67.4.2 queueInsertar() `void queueInsertar ( event_t nuevo )`

Agrega un evento a la cola.

Definition at line 53 of file queue.c.

4.67.4.3 queueSiguienteEvento() `event_t queueSiguienteEvento ( void )`

Devuelve el siguiente evento de la cola.

Returns

event_t

Definition at line 82 of file queue.c.

4.67.4.4 queueVacia() `bool queueVacia ( void )`

Chequea si la cola está vacía.

Returns

true Vacía

false No vacía

Definition at line 77 of file queue.c.

4.68 queue.h

Go to the documentation of this file.

```
00001
00012 // https://stackoverflow.com/questions/3536153/c-dynamically-growing-array
00013
00014 #ifndef _QUEUE_H_
00015 #define _QUEUE_H_
00016
00017 /********************************************************************************
00018  * INCLUDE HEADER FILES
00019  *******************************************************************************/
00020
00021 #include <stdbool.h>
00022
00023 /********************************************************************************
00024  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00025  *******************************************************************************/
00026
00027 // Tipo de dato para eventos
00028 typedef int event_t;
00029
00030 // Eventos posibles principales
00031 enum eventos
00032 {
00033     NADA = -1,
```

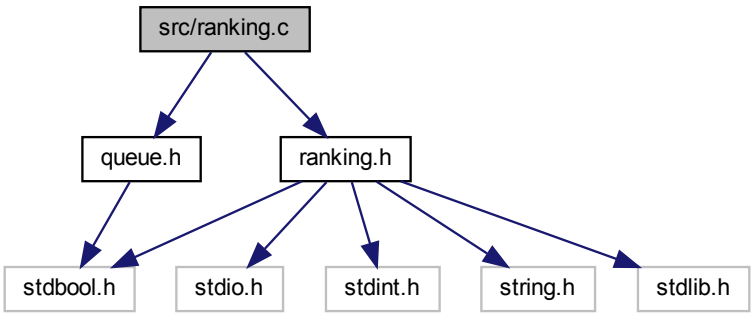
```
00034     SALIR = 0,
00035     GAME_OVER,
00036     FORCE_SALIR
00037 };
00038
00039 // Eventos posibles de interacción en el juego
00040 enum eventos_tecla
00041 {
00042     NO_MOVER = -1,
00043     ESC = 59,
00044     BORRAR = 63,
00045     ENTER = 67,
00046     IZDA = 82,
00047     DCHA,
00048     ARRIBA,
00049     ABAJO
00050 };
00051
00052 /*****************************************************************************
00053  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00054  *****************************************************************************/
00055
00060 void queueInsertar(event_t);
00061
00068 bool queueVacia(void);
00069
00075 event_t queueSiguienteEvento(void);
00076
00081 void destruirQueue(void);
00082
00083 /*****************************************************************************
00084  *****************************************************************************/
00085
00086 #endif // _QUEUE_H_
```

4.69 src/ranking.c File Reference

Source del modulo ranking. Funciones de interaccion con el ranking de jugadores. Permite trabajar con el txt correspondiente fácilmente,.

```
#include "ranking.h"
#include "queue.h"
```

Include dependency graph for ranking.c:



Macros

- #define MAX_LEN 100

Functions

- void iniciarRanking (void)

  Inicializa el sistema de ranking.
- void actualizarRanking (char ∗name, unsigned long long score)

  Actualiza el ranking de un jugador dado.
- void desiniciarRanking (void)

  Desinicializa el sistema de ranking, actualizando el archivo correspondiente.
- bool verificarJugadorRanking (char ∗name)

  Verifica si el jugador existe en el ranking.
- unsigned long long getJugadorRankingPuntos (char ∗name)

  Devuelve el puntaje de un jugador dado.
- uint getRankingLineas (void)

  Devuelve cantidad de renglones del ranking.
- char ∗∗ getRankingNombres (void)

  Devuelve array de nombres de jugadores.
- unsigned long long ∗ getRankingPuntos (void)

  Devuelve array de puntos de jugadores.

Variables

- FILE ∗ handlerRanking = NULL
- FILE ∗ handlerTemp = NULL

4.69.1 Detailed Description

Source del modulo ranking. Funciones de interaccion con el ranking de jugadores. Permite trabajar con el txt correspondiente fácilmente,.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file ranking.c.

4.69.2 Macro Definition Documentation

4.69.2.1 MAX_LEN `#define MAX_LEN 100`

Definition at line 25 of file ranking.c.

4.69.3 Function Documentation

4.69.3.1 actualizarRanking() `void actualizarRanking (`

```
char * name,
unsigned long long score )
```

Actualiza el ranking de un jugador dado.

Parameters

name	Nombre del jugador
score	Puntos del jugador

Definition at line 109 of file ranking.c.

4.69.3.2 desiniciarRanking() `void desiniciarRanking ( void )`

Desinicializa el sistema de ranking, actualizando el archivo correspondiente.

Definition at line 146 of file ranking.c.

4.69.3.3 getJugadorRankingPuntos() `unsigned long long getJugadorRankingPuntos ( char * name )`

Devuelve el puntaje de un jugador dado.

Parameters

name	Nombre del jugador

Returns

unsigned long long Score

Definition at line 173 of file ranking.c.

4.69.3.4 getRankingLineas() `uint getRankingLineas ( void )`

Devuelve cantidad de renglones del ranking.

Returns

int Renglones

Definition at line 193 of file ranking.c.

4.69.3.5 getRankingNombres() `char ** getRankingNombres ( void )`

Devuelve array de nombres de jugadores.

Returns

char**

Definition at line 198 of file ranking.c.

4.69.3.6 getRankingPuntos() `unsigned long long * getRankingPuntos ( void )`

Devuelve array de puntos de jugadores.

Returns

unsigned long long*

Definition at line 203 of file ranking.c.

4.69.3.7 iniciarRanking() `void iniciarRanking ( void )`

Inicializa el sistema de ranking.

Definition at line 95 of file ranking.c.

4.69.3.8 verificarJugadorRanking() `bool verificarJugadorRanking ( char * name )`

Verifica si el jugador existe en el ranking.

Parameters

name	Nombre del jugador

Returns

true Existe

false No existe

Definition at line 159 of file ranking.c.

4.69.4 Variable Documentation

4.69.4.1 handlerRanking `FILE* handlerRanking = NULL`

Definition at line 74 of file ranking.c.

4.69.4.2 handlerTemp `FILE* handlerTemp = NULL`

Definition at line 76 of file ranking.c.

4.70 ranking.c

Go to the documentation of this file.

```
00001
00013 /**********************************************************************************
00014  * INCLUDE HEADER FILES
00015  **********************************************************************************/
00016
00017 #include "ranking.h"
00018 #include "queue.h"
00019
00020 /**********************************************************************************
00021  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00022  **********************************************************************************/
00023
00024 // Largo maximo de una linea del txt
00025 #define MAX_LEN 100
00026
00027 /**********************************************************************************
00028  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00029  **********************************************************************************/
00030
00035 static void recargarRanking(void);
00036
00041 static void ordenarRanking(void);
00042
00047 static void writeRanking(void);
00048
00053 static void createRankingFile(void);
00054
00062 static void *realocar(void *p, size_t n);
00063
00064 /**********************************************************************************
00065  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00066  **********************************************************************************/
00067
00068 // Nombre del archivo de ranking
00069 static char *strRanking = "ranking.txt";
00070 // Nombre del archivo temporal
00071 static char *strTemp = "temp.txt";
00072
00073 // Handler del archivo de ranking
00074 FILE *handlerRanking = NULL;
00075 // Handler del archivo temporal
00076 FILE *handlerTemp = NULL;
00077
00078 // Punteros a nombres
00079 static char **names = NULL;
00080 // Puntero a scores
00081 static unsigned long long *scores = NULL;
00082
00083 // String temporal
00084 static char tempStr[MAX_LEN];
00085
00086 // Contador de lineas del txt
00087 static uint lineNumber = 0;
00088
00089 /**********************************************************************************
```

```
00090  ******************************************************************************
00091                          GLOBAL FUNCTION DEFINITIONS
00092  ******************************************************************************
00093  ******************************************************************************/
00094
00095 void iniciarRanking(void)
00096 {
00097     lineNumber = 0;
00098
00099     createRankingFile();
00100
00101     if ((handlerRanking = fopen(strRanking, "r")) == NULL)
00102         printf("Error opening ranking.txt");
00103
00104     recargarRanking();
00105
00106     fclose(handlerRanking);
00107 }
00108
00109 void actualizarRanking(char *name, unsigned long long score)
00110 {
00111     int i;
00112     bool player_exists = false;
00113
00114     if (lineNumber)
00115     {
00116         // Veo si el jugador esta en el ranking
00117         for (i = 0; i < lineNumber && !player_exists; i++)
00118         {
00119
00120             // Si el nombre coincide...
00121             if (strcmp(names[i], name) == 0)
00122             {
00123                 // Actualiza el score
00124                 scores[i] = score;
00125                 player_exists = true;
00126             }
00127         }
00128     }
00129
00130     // Si el jugador no existe en el ranking, lo agrego al final
00131     if (!player_exists)
00132     {
00133         names = (char **)realocar(names, sizeof(char *) * (lineNumber + 1));
00134         names[lineNumber] = (char *)realocar(NULL, strlen(name) * sizeof(char));
00135         strcpy(names[lineNumber], name);
00136
00137         scores = (unsigned long long *)realocar(scores, sizeof(unsigned long long) * (lineNumber +
      1));
00138         scores[lineNumber] = score;
00139
00140         lineNumber++;
00141     }
00142
00143     ordenarRanking();
00144 }
00145
00146 void desiniciarRanking(void)
00147 {
00148     // Escribe al archivo
00149     writeRanking();
00150
00151     // Liberacion de memoria
00152     int i;
00153     for (i = 0; i < lineNumber; i++)
00154         free(names[i]);
00155
00156     free(names);
00157 }
00158
00159 bool verificarJugadorRanking(char *name)
00160 {
00161     // Ranking vacio
00162     if (!lineNumber)
00163         return false;
00164
00165     int i;
00166     bool exists;
00167     for (i = 0, exists = false; i < lineNumber && !exists; i++)
00168         exists = strcmp(names[i], name) == 0;
00169
00170     return exists;
00171 }
00172
00173 unsigned long long getJugadorRankingPuntos(char *name)
00174 {
00175     int i;
```

```
00176     bool exists;
00177     unsigned long long score = 0;
00178 
00179     for (i = 0, exists = false; i < lineNumber && !exists; i++)
00180     {
00181         // Si el nombre coincide...
00182         if (strcmp(names[i], name) == 0)
00183         {
00184             // Carga el score
00185             score = scores[i];
00186             exists = true;
00187         }
00188     }
00189 
00190     return score;
00191 }
00192 
00193 uint getRankingLineas(void)
00194 {
00195     return lineNumber;
00196 }
00197 
00198 char **getRankingNombres(void)
00199 {
00200     return names;
00201 }
00202 
00203 unsigned long long *getRankingPuntos(void)
00204 {
00205     return scores;
00206 }
00207 
00208 /*****************************************************************************
00209  *****************************************************************************
00210                         LOCAL FUNCTION DEFINITIONS
00211  *****************************************************************************
00212  *****************************************************************************/
00213 
00214 static void recargarRanking(void)
00215 {
00216     lineNumber = 0;
00217 
00218     while (fgets(tempStr, MAX_LEN, handlerRanking) != NULL)
00219     {
00220         char *p = strchr(tempStr, '\n');
00221         while (p != NULL)
00222         {
00223             *p = '\0'; // Saco todos los saltos de linea
00224             p = strchr(tempStr, '\n');
00225         }
00226 
00227         char *tempPtr = strtok(tempStr, " ");                                       // Apunto al
      nombre
00228         names = (char **)realocar(names, sizeof(char *) * (lineNumber + 1));        // Reservo memoria
      para un puntero
00229         names[lineNumber] = (char *)realocar(NULL, strlen(tempPtr) * sizeof(char)); // Reservo memoria
      para el nombre
00230         strcpy(names[lineNumber], tempPtr);
00231 
00232         tempPtr = strtok(NULL, " ");
       // Apunto a los puntos
00233         scores = (unsigned long long *)realocar(scores, sizeof(unsigned long long) * (lineNumber +
      1)); // Reservo memoria para un score
00234         scores[lineNumber] = strtoul(tempPtr, NULL, 10);
00235 
00236         lineNumber++;
00237     }
00238 }
00239 
00240 static void ordenarRanking(void)
00241 {
00242     int i, j;
00243     unsigned long long tempScore;
00244 
00245     for (i = 0; i < (lineNumber - 1); i++)
00246     {
00247         for (j = 0; j < (lineNumber - i - 1); j++)
00248         {
00249             // Si el primer score es menor, o si es igual al siguiente pero predomina orden
      alfabetico...
00250             if ((scores[j] < scores[j + 1]) || ((scores[j] == scores[j + 1]) && (strcmp(names[j],
      names[j + 1]) > 0)))
00251             {
00252                 // Backup del menor
00253                 strcpy(tempStr, names[j]);
00254                 tempScore = scores[j];
00255 
```

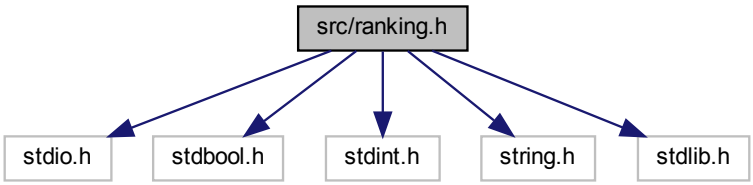
```
00256                 // El mayor se pone en la posicion del menor
00257                 strcpy(names[j], names[j + 1]);
00258                 scores[j] = scores[j + 1];
00259 
00260                 // El backup se pone en la posicion del mayor
00261                 strcpy(names[j + 1], tempStr);
00262                 scores[j + 1] = tempScore;
00263             }
00264         }
00265     }
00266 }
00267 
00268 static void writeRanking(void)
00269 {
00270     int i;
00271 
00272     // Crea archivo temporal
00273     if ((handlerTemp = fopen(strTemp, "w")) == NULL)
00274         printf("Error opening temp.txt");
00275 
00276     if (lineNumber)
00277     {
00278         // Copia lo nuevo en temp.txt
00279         for (i = 0; i < lineNumber; i++)
00280             fprintf(handlerTemp, "%s %lld\n", names[i], scores[i]);
00281     }
00282 
00283     remove(strRanking);
00284     rename(strTemp, strRanking);
00285 
00286     fclose(handlerTemp);
00287 }
00288 
00289 static void createRankingFile(void)
00290 {
00291     // crea el archivo, si no lo estaba
00292     FILE *pFile;
00293     if ((pFile = fopen(strRanking, "a")) == NULL)
00294     {
00295         printf("Error creando %s", strRanking);
00296     }
00297     fclose(pFile);
00298 }
00299 
00300 static void *realocar(void *p, size_t n)
00301 {
00302     void *aux = realloc(p, n);
00303     if (aux == NULL)
00304     {
00305         perror("Error en ranking.c al realocar memoria\n");
00306         free(p);
00307         queueInsertar(FORCE_SALIR);
00308     }
00309     return aux;
00310 }
```

4.71 src/ranking.h File Reference

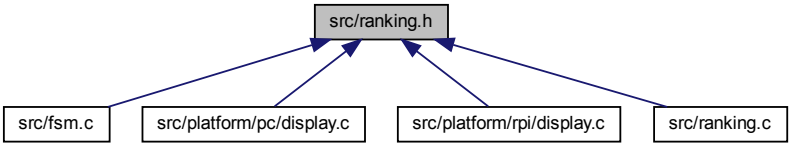
Header del modulo ranking. Prototipos de funciones de interaccion con el ranking de jugadores. Permite trabajar con el txt correspondiente fácilmente,.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for ranking.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define DEFAULT_PLAYER_NAME "PLAYER"

Functions

- void iniciarRanking (void)

  Inicializa el sistema de ranking.
- void actualizarRanking (char ∗name, unsigned long long score)

  Actualiza el ranking de un jugador dado.
- void desiniciarRanking (void)

  Desinicializa el sistema de ranking, actualizando el archivo correspondiente.
- bool verificarJugadorRanking (char ∗name)

  Verifica si el jugador existe en el ranking.
- unsigned long long getJugadorRankingPuntos (char ∗name)

  Devuelve el puntaje de un jugador dado.
- uint getRankingLineas (void)

  Devuelve cantidad de renglones del ranking.
- char ∗∗ getRankingNombres (void)

  Devuelve array de nombres de jugadores.
- unsigned long long ∗ getRankingPuntos (void)

  Devuelve array de puntos de jugadores.

4.71.1 Detailed Description

Header del modulo ranking. Prototipos de funciones de interaccion con el ranking de jugadores. Permite trabajar con el txt correspondiente fácilmente,.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file ranking.h.

4.71.2 Macro Definition Documentation

4.71.2.1 DEFAULT_PLAYER_NAME `#define DEFAULT_PLAYER_NAME "PLAYER"`

Definition at line 30 of file ranking.h.

4.71.3 Function Documentation

4.71.3.1 actualizarRanking() `void actualizarRanking (`

```
char * name,
unsigned long long score )
```

Actualiza el ranking de un jugador dado.

Parameters

name	Nombre del jugador
score	Puntos del jugador

Definition at line 109 of file ranking.c.

4.71.3.2 desiniciarRanking() `void desiniciarRanking (`

```
void  )
```

Desinicializa el sistema de ranking, actualizando el archivo correspondiente.

Definition at line 146 of file ranking.c.

4.71.3.3 getJugadorRankingPuntos() `unsigned long long getJugadorRankingPuntos ( char * name )`

Devuelve el puntaje de un jugador dado.

Parameters

name	Nombre del jugador

Returns

unsigned long long Score

Definition at line 173 of file ranking.c.

4.71.3.4 getRankingLineas() `uint getRankingLineas ( void )`

Devuelve cantidad de renglones del ranking.

Returns

int Renglones

Definition at line 193 of file ranking.c.

4.71.3.5 getRankingNombres() `char ** getRankingNombres ( void )`

Devuelve array de nombres de jugadores.

Returns

char∗∗

Definition at line 198 of file ranking.c.

4.71.3.6 getRankingPuntos() `unsigned long long * getRankingPuntos ( void )`

Devuelve array de puntos de jugadores.

Returns

unsigned long long∗

Definition at line 203 of file ranking.c.

4.71.3.7 iniciarRanking() void iniciarRanking (void)

Inicializa el sistema de ranking.

Definition at line 95 of file ranking.c.

4.71.3.8 verificarJugadorRanking() bool verificarJugadorRanking (char * name)

Verifica si el jugador existe en el ranking.

Parameters

name	Nombre del jugador

Returns

true Existe

false No existe

Definition at line 159 of file ranking.c.

4.72 ranking.h

Go to the documentation of this file.

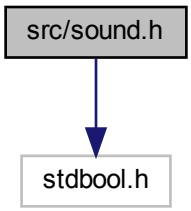
```
00001
00013 #ifndef _RANKING_H_
00014 #define _RANKING_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdio.h>
00021 #include <stdbool.h>
00022 #include <stdint.h>
00023 #include <string.h>
00024 #include <stdlib.h>
00025
00026 /*****************************************************************************
00027  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00028  *****************************************************************************/
00029
00030 #define DEFAULT_PLAYER_NAME "PLAYER"
00031
00032 /*****************************************************************************
00033  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00034  *****************************************************************************/
00035
00040 void iniciarRanking(void);
00041
00048 void actualizarRanking(char *name, unsigned long long score);
00049
00054 void desiniciarRanking(void);
00055
00063 bool verificarJugadorRanking(char *name);
00064
00071 unsigned long long getJugadorRankingPuntos(char *name);
00072
00078 uint getRankingLineas(void);
00079
00085 char **getRankingNombres(void);
```

```
00086
00092 unsigned long long *getRankingPuntos(void);
00093
00094 /*****************************************************************************
00095  *****************************************************************************/
00096
00097 #endif // _RANKING_H_
```

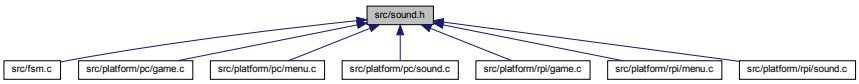
4.73 src/sound.h File Reference

Header del modulo sound Vinculo entre la fsm y las plataformas en lo que respecta al sonido.

`#include <stdbool.h>`
Include dependency graph for sound.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum **musica** {
  MUSICA_MENU_PPAL , **MUSICA_RANKING** , **MUSICA_CREDITOS** , **MUSICA_JUGANDO** ,
  MUSICA_MENU_PAUSA , **MUSICA_GAME_OVER** , **SIZEOF_MUSICA** }
- enum **efectos** {
  EFECTO_SELECCION , **EFECTO_SALTO** , **EFECTO_IMPACTO** , **EFECTO_AHOGADO** ,
  EFECTO_POCO_TIEMPO , **EFECTO_META** , **EFECTO_NIVEL_COMPLETO** , **EFECTO_NUEVO_MAX_↩**
  SCORE ,
  EFECTO_MENU_ENTER , **EFECTO_SALIENDO** , **EFECTO_SIN_TIEMPO** , **SIZEOF_EFECTOS** }

Functions

- bool iniciarSonido (void)

  Inicializa el sonido de la plataforma.
- void destruirSonido (void)

  Desinicializa el sonido de la plataforma.
- void pausarMusica (void)

  Pausa la musica actual.
- void reproducirMusica (int musica)

  Pone a reproducir una musica dada.
- void reproducirEfecto (int efecto)

  Pone a reproducir un efecto dado.

4.73.1 Detailed Description

Header del modulo sound Vinculo entre la fsm y las plataformas en lo que respecta al sonido.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file sound.h.

4.73.2 Enumeration Type Documentation

4.73.2.1 efectos `enum efectos`

Definition at line 39 of file sound.h.

4.73.2.2 musica `enum musica`

Definition at line 27 of file sound.h.

4.73.3 Function Documentation

4.73.3.1 destruirSonido() `void destruirSonido ( void )`

Desinicializa el sonido de la plataforma.

Definition at line 32 of file sound.c.

4.73.3.2 iniciarSonido() `bool iniciarSonido ( void )`

Inicializa el sonido de la plataforma.

Returns

true Exito

false Error

Definition at line 27 of file sound.c.

4.73.3.3 pausarMusica() `void pausarMusica ( void )`

Pausa la musica actual.

Definition at line 36 of file sound.c.

4.73.3.4 reproducirEfecto() `void reproducirEfecto ( int efecto )`

Pone a reproducir un efecto dado.

Parameters

int	num efectos

Definition at line 76 of file sound.c.

4.73.3.5 reproducirMusica() `void reproducirMusica ( int musica )`

Pone a reproducir una musica dada.

Parameters

int	enum musica

Definition at line 41 of file sound.c.

4.74 sound.h

Go to the documentation of this file.

```
00001
00013 #ifndef _SOUND_H_
00014 #define _SOUND_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdbool.h>
00021
00022 /*****************************************************************************
00023  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00024  *****************************************************************************/
00025
00026 // Musicas a usar
00027 enum musica
00028 {
00029     MUSICA_MENU_PPAL,
00030     MUSICA_RANKING,
00031     MUSICA_CREDITOS,
00032     MUSICA_JUGANDO,
00033     MUSICA_MENU_PAUSA,
00034     MUSICA_GAME_OVER,
00035     SIZEOF_MUSICA
00036 };
00037
00038 // Efectos a usar
00039 enum efectos
00040 {
00041     EFECTO_SELECCION,
00042     EFECTO_SALTO,
00043     EFECTO_IMPACTO,
00044     EFECTO_AHOGADO,
00045     EFECTO_POCO_TIEMPO,
00046     EFECTO_META,
00047     EFECTO_NIVEL_COMPLETO,
00048     EFECTO_NUEVO_MAX_SCORE,
00049     EFECTO_MENU_ENTER,
00050     EFECTO_SALIENDO,
00051     EFECTO_SIN_TIEMPO,
00052     SIZEOF_EFECTOS
00053 };
00054
00055 /*****************************************************************************
00056  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00057  *****************************************************************************/
00058
00065 bool iniciarSonido(void);
00066
00071 void destruirSonido(void);
00072
00077 void pausarMusica(void);
00078
00084 void reproducirMusica(int musica);
00085
00091 void reproducirEfecto(int efecto);
00092
00093 /*****************************************************************************
00094  *****************************************************************************/
00095
00096 #endif // _SOUND_H_
```

Index