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 89 of file allegro_stuff.c.

3.1.2 Field Documentation

3.1.2.1 disp `ALLEGRO_DISPLAY* disp`

Definition at line 98 of file allegro_stuff.c.

3.1.2.2 done `bool done`

Definition at line 109 of file allegro_stuff.c.

3.1.2.3 event `ALLEGRO_EVENT event`

Definition at line 106 of file allegro_stuff.c.

3.1.2.4 font ALLEGRO_FONT* font

Definition at line 101 of file allegro_stuff.c.

3.1.2.5 font_h int font_h

Definition at line 102 of file allegro_stuff.c.

3.1.2.6 font_w int font_w

Definition at line 103 of file allegro_stuff.c.

3.1.2.7 queue ALLEGRO_EVENT_QUEUE* queue

Definition at line 95 of file allegro_stuff.c.

3.1.2.8 redraw bool redraw

Definition at line 111 of file allegro_stuff.c.

3.1.2.9 timer ALLEGRO_TIMER* timer

Definition at line 92 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 90 of file entities.c.

3.2.2 Field Documentation

3.2.2.1 count `int count`

Definition at line 98 of file entities.c.

3.2.2.2 dx `int dx`

Definition at line 95 of file entities.c.

3.2.2.3 fast `bool fast`

Definition at line 99 of file entities.c.

3.2.2.4 lane `int lane`

Definition at line 94 of file entities.c.

3.2.2.5 length `int length`

Definition at line 97 of file entities.c.

3.2.2.6 type `CAR_TYPE type`

Definition at line 96 of file entities.c.

3.2.2.7 used `bool used`

Definition at line 100 of file entities.c.

3.2.2.8 x `int x`

Definition at line 92 of file entities.c.

3.2.2.9 y `int y`

Definition at line 93 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 133 of file entities.c.

3.3.2 Field Documentation

3.3.2.1 blink_timer `unsigned int blink_timer`

Definition at line 142 of file entities.c.

3.3.2.2 cont `unsigned int cont`

Definition at line 143 of file entities.c.

3.3.2.3 flag `bool flag`

Definition at line 144 of file entities.c.

3.3.2.4 frame_cont `unsigned int frame_cont`

Definition at line 140 of file entities.c.

3.3.2.5 timeout `unsigned int timeout`

Definition at line 141 of file entities.c.

3.3.2.6 used `bool used`

Definition at line 137 of file entities.c.

3.3.2.7 x `int x`

Definition at line 135 of file entities.c.

3.3.2.8 y `int y`

Definition at line 136 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 52 of file game_data.c.

3.4.2 Field Documentation

3.4.2.1 difficulty `int difficulty`

Definition at line 69 of file game_data.c.

3.4.2.2 flag `unsigned char flag`

Definition at line 73 of file game_data.c.

3.4.2.3 frames `unsigned long frames`

Definition at line 66 of file game_data.c.

3.4.2.4 goals `bool goals[MAX_GOALS]`

Definition at line 75 of file game_data.c.

3.4.2.5 lives `int lives`

Definition at line 54 of file game_data.c.

3.4.2.6 name `char name[MAX_NAME_CHAR]`

Definition at line 71 of file game_data.c.

3.4.2.7 number `int number`

Definition at line 60 of file game_data.c.

3.4.2.8 score `unsigned long long score`

Definition at line 55 of file game_data.c.

3.4.2.9 score_max `unsigned long long score_max`

Definition at line 56 of file game_data.c.

3.4.2.10 time `int time`

Definition at line 62 of file game_data.c.

3.4.2.11 time_left `int time_left`

Definition at line 61 of file game_data.c.

3.4.2.12 time_ref `long time_ref`

Definition at line 63 of file game_data.c.

3.4.2.13 timer_in_sec `int timer_in_sec`

Definition at line 67 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 78 of file entities.c.

3.6.2 Field Documentation

3.6.2.1 facing `int facing`

Definition at line 83 of file entities.c.

3.6.2.2 moving `int moving`

Definition at line 82 of file entities.c.

3.6.2.3 next_action `unsigned char next_action`

Definition at line 86 of file entities.c.

3.6.2.4 state `unsigned char state`

Definition at line 85 of file entities.c.

3.6.2.5 steps `int steps`

Definition at line 84 of file entities.c.

3.6.2.6 x int x

Definition at line 80 of file entities.c.

3.6.2.7 y int y

Definition at line 81 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 103 of file entities.c.

3.8.2 Field Documentation

3.8.2.1 dx `int dx`

Definition at line 108 of file entities.c.

3.8.2.2 lane `int lane`

Definition at line 107 of file entities.c.

3.8.2.3 used `bool used`

Definition at line 109 of file entities.c.

3.8.2.4 x `int x`

Definition at line 105 of file entities.c.

3.8.2.5 y `int y`

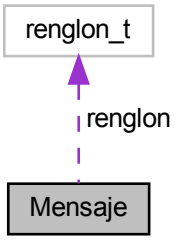
Definition at line 106 of file entities.c.

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`

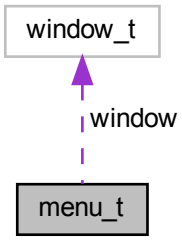
Definition at line 53 of file mensajes.h.

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]`

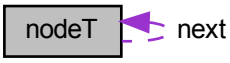
Definition at line 43 of file menu.c.

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 115 of file allegro_stuff.c.

3.14.2 Field Documentation

3.14.2.1 bonus `ALLEGRO_SAMPLE* bonus`

Definition at line 127 of file allegro_stuff.c.

3.14.2.2 click ALLEGRO_SAMPLE* click

Definition at line 126 of file allegro_stuff.c.

3.14.2.3 coin_drop ALLEGRO_SAMPLE* coin_drop

Definition at line 134 of file allegro_stuff.c.

3.14.2.4 crash ALLEGRO_SAMPLE* crash

Definition at line 123 of file allegro_stuff.c.

3.14.2.5 drowned ALLEGRO_SAMPLE* drowned

Definition at line 129 of file allegro_stuff.c.

3.14.2.6 exiting ALLEGRO_SAMPLE* exiting

Definition at line 132 of file allegro_stuff.c.

3.14.2.7 goal ALLEGRO_SAMPLE* goal

Definition at line 124 of file allegro_stuff.c.

3.14.2.8 jump ALLEGRO_SAMPLE* jump

Definition at line 122 of file allegro_stuff.c.

3.14.2.9 low_time ALLEGRO_SAMPLE* low_time

Definition at line 125 of file allegro_stuff.c.

3.14.2.10 menu_enter ALLEGRO_SAMPLE* menu_enter

Definition at line 130 of file allegro_stuff.c.

3.14.2.11 new_max_score ALLEGRO_SAMPLE* new_max_score

Definition at line 131 of file allegro_stuff.c.

3.14.2.12 no_time ALLEGRO_SAMPLE* no_time

Definition at line 133 of file allegro_stuff.c.

3.14.2.13 run_completed ALLEGRO_SAMPLE* run_completed

Definition at line 128 of file allegro_stuff.c.

3.14.2.14 stream ALLEGRO_AUDIO_STREAM* stream

Definition at line 117 of file allegro_stuff.c.

3.14.2.15 stream_state unsigned char stream_state

Definition at line 118 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 * tutorial
- 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 105 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 91 of file allegro_stuff.h.

3.16.2.7 frame `ALLEGRO_BITMAP* frame[SPRITE_SPLASH_FRAMES]`

Definition at line 96 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 89 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 tutorial `ALLEGRO_BITMAP* tutorial`

Definition at line 87 of file allegro_stuff.h.

3.16.2.18 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 113 of file entities.c.

3.18.2 Field Documentation

3.18.2.1 cont `unsigned int cont`

Definition at line 126 of file entities.c.

3.18.2.2 dx `int dx`

Definition at line 118 of file entities.c.

3.18.2.3 frame `unsigned char frame`

Definition at line 124 of file entities.c.

3.18.2.4 lane `int lane`

Definition at line 117 of file entities.c.

3.18.2.5 state `unsigned char state`

Definition at line 130 of file entities.c.

3.18.2.6 timeout `unsigned int timeout`

Definition at line 125 of file entities.c.

3.18.2.7 turtles_in_pack `unsigned char turtles_in_pack`

Definition at line 120 of file entities.c.

3.18.2.8 used `bool used`

Definition at line 119 of file entities.c.

3.18.2.9 wide `int wide`

Definition at line 129 of file entities.c.

3.18.2.10 x `int x`

Definition at line 115 of file entities.c.

3.18.2.11 y `int y`

Definition at line 116 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:

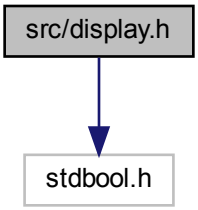
- src/platform/pc/menu.c

4 File Documentation

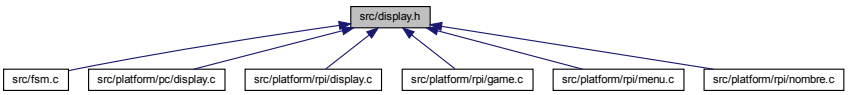
4.1 src/display.h File Reference

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_NOMBRE** , **POS_MSJ_PASAR** ,
  POS_MSJ_PAUSA , **POS_MSJ_NEW_HI_SCORE** , **POS_MSJ_GAME_OVER** , **POS_OPCION** ,
  POS_RANKING_2 , **POS_CREDITOS_INTRO** , **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.

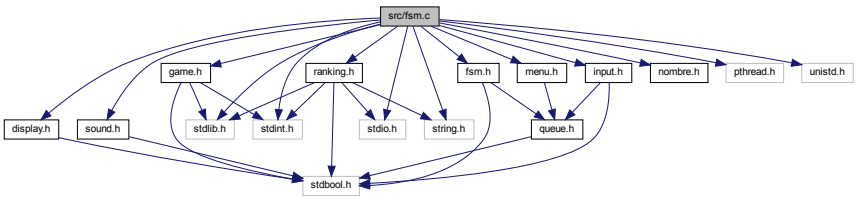
```
00001
00013 #ifndef _DISPLAY_H_
00014 #define _DISPLAY_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdbool.h>
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_NOMBRE,
00032     POS_MSJ_PASAR,
00033     POS_MSJ_PAUSA,
00034     POS_MSJ_NEW_HI_SCORE,
00035     POS_MSJ_GAME_OVER,
00036     POS_OPCION,
00037     POS_RANKING_2,
00038   POS_CREDITOS_INTRO,
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 seleccionando_dificultad [ ]
- STATE viendo_ranking [ ]
- STATE viendo_creditos [ ]
- STATE poniendo_nombre [ ]
- STATE jugando [ ]
- STATE en_pausa [ ]
- STATE en_game_over [ ]

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 337 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 319 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 299 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, procesar_enter_menu},
        {ARRIBA, en_game_over, subirOpcion},
        {ABAJO, en_game_over, bajarOpcion},
        {FORCE_SALIR, NULL, salir_del_juego},
        {CTE_OPCION, jugando, iniciar_juego},
        {CTE_OPCION + 1, en_menu_ppal, ir_a_menu_ppal},
        {FIN_TABLA, en_game_over, do_nothing}}
```

Definition at line 281 of file fsm.c.

4.3.5.2 en_menu_ppal STATE en_menu_ppal

Initial value:

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

Definition at line 215 of file fsm.c.

4.3.5.3 en_pausa STATE en_pausa

Initial value:

```
=
    {
        {ENTER, en_pausa, procesar_enter_menu},
        {ARRIBA, en_pausa, subirOpcion},
        {ABAJO, en_pausa, bajarOpcion},
        {FORCE_SALIR, NULL, salir_del_juego},
        {CTE_OPCION, jugando, continuar},
        {CTE_OPCION + 1, jugando, iniciar_juego},
        {CTE_OPCION + 2, en_menu_ppal, ir_a_menu_ppal},
        {FIN_TABLA, en_pausa, do_nothing}}
```

Definition at line 270 of file fsm.c.

4.3.5.4 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 259 of file fsm.c.

4.3.5.5 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 248 of file fsm.c.

4.3.5.6 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 228 of file fsm.c.

4.3.5.7 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 242 of file fsm.c.

4.3.5.8 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 236 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 una arista
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 seleccionando_dificultad[];
00207 extern STATE viendo_ranking[];
00208 extern STATE viendo_creditos[];
00209 extern STATE poniendo_nombre[];
00210 extern STATE jugando[];
00211 extern STATE en_pausa[];
00212 extern STATE en_game_over[];
00213 // Forward declarations de los estados
00214
00215 STATE en_menu_ppal[] =
00216     {
00217         {ENTER, en_menu_ppal, procesar_enter_menu},
00218         {ARRIBA, en_menu_ppal, subirOpcion},
00219         {ABAJO, en_menu_ppal, bajarOpcion},
00220         {FORCE_SALIR, NULL, salir_del_juego},
00221         {CTE_OPCION, poniendo_nombre, ir_a_poniendo_nombre},
00222         {CTE_OPCION + 1, seleccionando_dificultad, ir_a_seleccionando_dificultad},
00223         {CTE_OPCION + 2, viendo_ranking, ir_a_viendo_ranking},
00224         {CTE_OPCION + 3, viendo_creditos, ir_a_viendo_creditos},
00225         {CTE_OPCION + 4, NULL, salir_del_juego},
00226         {FIN_TABLA, en_menu_ppal, do_nothing}};
00227
00228 STATE seleccionando_dificultad[] =
00229     {
00230         {ENTER, en_menu_ppal, procesar_enter_dificultad},
00231         {ARRIBA, seleccionando_dificultad, subirOpcion},
00232         {ABAJO, seleccionando_dificultad, bajarOpcion},
00233         {FORCE_SALIR, NULL, salir_del_juego},
00234         {FIN_TABLA, seleccionando_dificultad, do_nothing}};
00235
00236 STATE viendo_ranking[] =
00237     {
00238         {ENTER, en_menu_ppal, procesar_enter_ranking},
00239         {FORCE_SALIR, NULL, salir_del_juego},
00240         {FIN_TABLA, viendo_ranking, do_nothing}};
00241
00242 STATE viendo_creditos[] =
00243     {
00244         {ENTER, en_menu_ppal, procesar_enter_creditos},
00245         {FORCE_SALIR, NULL, salir_del_juego},
00246         {FIN_TABLA, viendo_creditos, do_nothing}};
00247
00248 STATE poniendo_nombre[] =
00249     {
00250         {ESC, en_menu_ppal, ir_a_menu_ppal},
00251         {ENTER, jugando, iniciar_juego},
00252         {ARRIBA, poniendo_nombre, subirLetra},
00253         {ABAJO, poniendo_nombre, bajarLetra},
00254         {DCHA, poniendo_nombre, siguienteLetra},
00255         {FORCE_SALIR, NULL, salir_del_juego},
00256         {FIN_TABLA, poniendo_nombre, agregarLetra} // Si no coincide el evento con ninguna de las
      teclas previas, se toam como si se apretase una letra
00257 };
00258
00259 STATE jugando[] =
00260     {
00261         {ENTER, en_pausa, pausar},
00262         {GAME_OVER, en_game_over, procesar_game_over},
00263         {ARRIBA, jugando, moverAdelante},
```

```
00264         {ABAJO, jugando, moverAtras},
00265         {IZDA, jugando, moverIzda},
00266         {DCHA, jugando, moverDcha},
00267         {FORCE_SALIR, NULL, salir_del_juego},
00268         {FIN_TABLA, jugando, do_nothing}};
00269
00270 STATE en_pausa[] =
00271     {
00272         {ENTER, en_pausa, procesar_enter_menu},
00273         {ARRIBA, en_pausa, subirOpcion},
00274         {ABAJO, en_pausa, bajarOpcion},
00275         {FORCE_SALIR, NULL, salir_del_juego},
00276         {CTE_OPCION, jugando, continuar},
00277         {CTE_OPCION + 1, jugando, iniciar_juego},
00278         {CTE_OPCION + 2, en_menu_ppal, ir_a_menu_ppal},
00279         {FIN_TABLA, en_pausa, do_nothing}};
00280
00281 STATE en_game_over[] =
00282     {
00283         {ENTER, en_game_over, procesar_enter_menu},
00284         {ARRIBA, en_game_over, subirOpcion},
00285         {ABAJO, en_game_over, bajarOpcion},
00286         {FORCE_SALIR, NULL, salir_del_juego},
00287         {CTE_OPCION, jugando, iniciar_juego},
00288         {CTE_OPCION + 1, en_menu_ppal, ir_a_menu_ppal},
00289         {FIN_TABLA, en_game_over, do_nothing}};
00290
00291 #pragma endregion FSM STATES8
00292
00293 /*****************************************************************************
00294  *****************************************************************************
00295                         GLOBAL FUNCTION DEFINITIONS
00296  *****************************************************************************
00297  *****************************************************************************/
00298
00299 bool inicializarFsm(void)
00300 {
00301   p2CurrentState = en_menu_ppal;
00302
00303   srand(time(NULL));
00304
00305   iniciarDisplay();
00306   iniciarMenu();
00307   iniciarEntradas();
00308   iniciarSonido();
00309
00310   iniciarRanking();
00311
00312   ir_a_menu_ppal();
00313
00314   pthread_create(&tinput, NULL, threadInput, NULL);
00315
00316   return true;
00317 }
00318
00319 void fsm(event_t evento_actual)
00320 {
00321   STATE *aux = p2CurrentState;
00322   /*
00323     Mientras el evento actual no coincida con uno "interesante", y mientras no se haya recorrido
00324     todo el estado...
00325     */
00326   while ((aux->evento != evento_actual) && (aux->evento != FIN_TABLA))
00327     // Verifico con la siguiente posibilidad dentro del mismo estado.
00328     ++aux;
00329
00330   // Pasa al siguiente estado
00331   p2CurrentState = aux->proximo_estado;
00332
00333   // Ejecuta la rutina correspondiente
00334   (*aux->p_rut_accion)();
00335 }
00336
00337 void fixHighCpuUsage(void)
00338 {
00339   usleep(FIX_CPU_USAGE_SLEEP_US);
00340 }
00341
00342 /*****************************************************************************
00343  *****************************************************************************
00344                         LOCAL FUNCTION DEFINITIONS
00345  *****************************************************************************
00346  *****************************************************************************/
00347
00348 static void *threadInput(void *ptr)
00349 {
00350   while (p2CurrentState)
```

```
00351   {
00352     event_t entrada = leerEntradas();
00353     if (entrada != NADA)
00354       queueInsertar(entrada);
00355 
00356     fixHighCpuUsage();
00357   }
00358 
00359   return NULL;
00360 }
00361 
00362 static void *threadJuego(void *ptr)
00363 {
00364 
00365   reconfigurarDisplayON();
00366 
00367   srand(time(NULL));
00368 
00369   while (p2CurrentState == jugando)
00370   {
00371     if (tiempoRefrescoEntidades())
00372       refrescar();
00373 
00374     actualizarInterfaz();
00375 
00376     fixHighCpuUsage();
00377   }
00378 
00379   pausarJuego();
00380 
00381   reconfigurarDisplayOFF();
00382 
00383   return NULL;
00384 }
00385 
00386 static void *threadDisplayRanking(void *ptr)
00387 {
00388   reconfigurarDisplayON();
00389 
00390   cargarRanking();
00391 
00392   while (p2CurrentState == viendo_ranking)
00393     mostrarRanking();
00394 
00395   limpiarDisplay();
00396 
00397   reconfigurarDisplayOFF();
00398 
00399   return NULL;
00400 }
00401 
00402 static void *threadDisplayCreditos(void *ptr)
00403 {
00404   reconfigurarDisplayON();
00405 
00406   cargarCreditos();
00407 
00408   while (p2CurrentState == viendo_creditos)
00409   {
00410     mostrarCreditos();
00411     fixHighCpuUsage();
00412   }
00413 
00414   reconfigurarDisplayOFF();
00415 
00416   return NULL;
00417 }
00418 
00419 static void do_nothing(void)
00420 {
00421 }
00422 
00423 static void procesar_enter_menu(void)
00424 {
00425   reproducirEfecto(EFECTO_MENU_ENTER);
00426   queueInsertar(CTE_OPCION + getOpcion());
00427 }
00428 
00429 static void ir_a_menu_ppal()
00430 {
00431   limpiarDisplay();
00432   dejarTexto("MENU", POS_MSJ_MENU, true);
00433   reproducirMusica(MUSICA_MENU_PPAL);
00434   int menu[5] = {JUGAR, DIFICULTAD, RANKING, CREDITOS, SALIRTXT};
00435   setMenu(menu, 5);
00436   setOpcion(0);
00437 }
```

```
00438
00439 static void ir_a_viendo_ranking()
00440 {
00441   limpiarDisplay();
00442   reconfigurarDisplayOFF();
00443   reproducirMusica(MUSICA_RANKING);
00444   pthread_create(&tdisplayranking, NULL, threadDisplayRanking, NULL);
00445 }
00446
00447 static void ir_a_viendo_creditos(void)
00448 {
00449   limpiarDisplay();
00450   reconfigurarDisplayOFF();
00451   reproducirMusica(MUSICA_CREDITOS);
00452   pthread_create(&tdisplaycreditos, NULL, threadDisplayCreditos, NULL);
00453 }
00454
00455 static void salir_del_juego()
00456 {
00457   pthread_join(tinput, NULL);
00458   reproducirEfecto(EFECTO_SALIENDO);
00459   sleep(2);
00460   destruirMenu();
00461   destruirSonido();
00462   desiniciarRanking();
00463   limpiarDisplay();
00464   queueInsertar(SALIR);
00465 }
00466
00467 static void procesar_enter_ranking(void)
00468 {
00469   pthread_join(tdisplayranking, NULL);
00470   reconfigurarDisplayON();
00471   ir_a_menu_ppal();
00472 }
00473
00474 static void procesar_enter_creditos(void)
00475 {
00476   pthread_join(tdisplaycreditos, NULL);
00477   reconfigurarDisplayON();
00478   ir_a_menu_ppal();
00479 }
00480
00481 static void iniciar_juego(void)
00482 {
00483   limpiarDisplay();
00484   char *nombreJugador = devolverNombre();
00485   if (nombreJugador == NULL)
00486     setNombre(DEFAULT_PLAYER_NAME);
00487   else if (nombreJugador[0] == 0)
00488     setNombre(DEFAULT_PLAYER_NAME);
00489   else
00490     setNombre(nombreJugador);
00491
00492   if (verificarJugadorRanking(getNombre()))
00493     setMaxPuntos(getJugadorRankingPuntos(getNombre()));
00494
00495   inicializarJuego();
00496   reconfigurarDisplayOFF();
00497
00498   reproducirMusica(MUSICA_JUGANDO);
00499
00500   reiniciarNivel();
00501   pthread_create(&tjuego, NULL, threadJuego, NULL);
00502 }
00503
00504 static void ir_a_poniendo_nombre()
00505 {
00506   limpiarDisplay();
00507   dejarTexto("INGRESE NOMBRE", POS_MSJ_NOMBRE, true);
00508   nuevoNombre();
00509 }
00510
00511 static void ir_a_seleccionando_dificultad()
00512 {
00513   limpiarDisplay();
00514   dejarTexto("DIFICULTAD", POS_MSJ_DIFICULTAD, true);
00515   int menu[3] = {FACIL, NORMAL, DIFICIL};
00516   setMenu(menu, 3);
00517   setOpcion(0);
00518 }
00519
00520 static void procesar_enter_dificultad(void)
00521 {
00522   setDificultad(getOpcion());
00523   reproducirEfecto(EFECTO_MENU_ENTER);
00524   ir_a_menu_ppal();
```

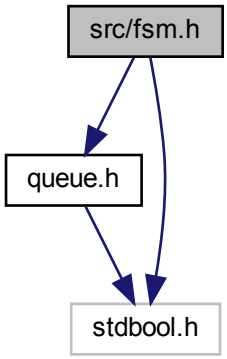
```
00525 }
00526
00527 static void pausar(void)
00528 {
00529   limpiarDisplay();
00530   pthread_join(tjuego, NULL);
00531   reproducirMusica(MUSICA_MENU_PAUSA);
00532   reconfigurarDisplayON();
00533   dejarTexto("PAUSA", POS_MSJ_PAUSA, true);
00534   int menu[3] = {CONTINUAR, REINICIAR, SALIRTXT};
00535   setMenu(menu, 3);
00536   setOpcion(0);
00537 }
00538
00539 static void continuar(void)
00540 {
00541   limpiarDisplay();
00542   reconfigurarDisplayOFF();
00543   reproducirMusica(MUSICA_JUGANDO);
00544   reanudarJuego();
00545   pthread_create(&tjuego, NULL, threadJuego, NULL);
00546 }
00547
00548 static void procesar_game_over(void)
00549 {
00550   pthread_join(tjuego, NULL);
00551
00552   limpiarDisplay();
00553
00554   reproducirMusica(MUSICA_GAME_OVER);
00555   reconfigurarDisplayON();
00556
00557   unsigned long long jugador_puntos = getPuntos();
00558
00559   if (jugador_puntos > getMaxPuntos())
00560   {
00561     reproducirEfecto(EFECTO_NUEVO_MAX_SCORE);
00562
00563     mostrarTexto("NUEVA PUNTUACION ALTA", POS_MSJ_NEW_HI_SCORE);
00564     setMaxPuntos(jugador_puntos);
00565
00566     actualizarRanking(getNombre(), getMaxPuntos());
00567   }
00568   limpiarDisplay();
00569   dejarTexto("FIN DEL JUEGO", POS_MSJ_GAME_OVER, true);
00570   int menu[2] = {REINICIAR, SALIRTXT};
00571   setMenu(menu, 2);
00572   setOpcion(0);
00573 }
```

4.5 src/fsm.h File Reference

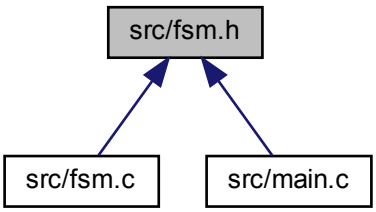
Header del modulo fsm. Contiene los prototipos de funciones necesarias para iniciar la fsm desde main.c.

```
#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

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 337 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 319 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 299 of file fsm.c.

4.6 fsm.h

Go to the documentation of this file.

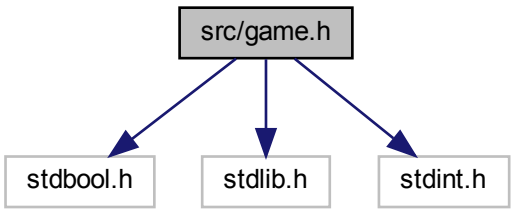
```
00001
00013 #ifndef _FSM_H_
00014 #define _FSM_H_
00015
00016 /*******************************************************************************
00017  * INCLUDE HEADER FILES
00018  ******************************************************************************/
00019
00020 #include "queue.h"
00021 #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

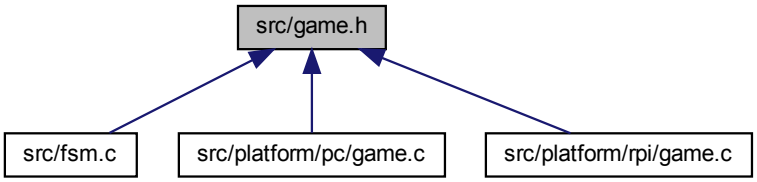
Header del modulo game Vinculo entre la fsm y las plataformas en lo que respecta a la informacion del jugador y el progreso del juego.

```
#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 302 of file game.c.

4.7.2.13 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 197 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);
```

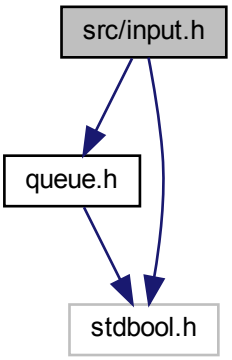
```
00041
00047 void setDificultad(int dif);
00048
00055 bool tiempoRefrescoEntidades(void);
00056
00062 char *getNombre(void);
00063
00069 unsigned long long getPuntos(void);
00070
00076 unsigned long long getMaxPuntos(void);
00077
00083 int getNivel(void);
00084
00089 void inicializarJuego(void);
00090
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

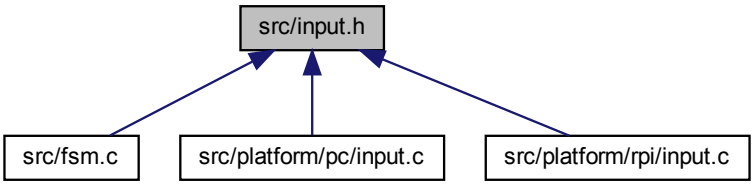
Header del modulo input Vinculo entre la fsm y las plataformas en lo que respecta al manejo de acciones externas.

```
#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

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

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.

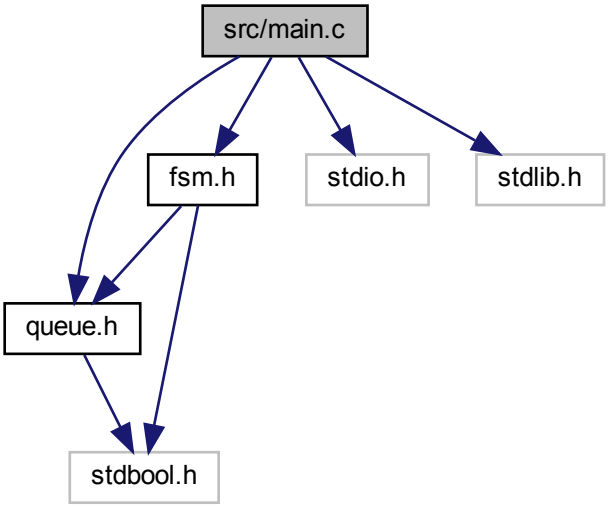
```
00001
00013 #ifndef _INPUT_H_
00014 #define _INPUT_H_
00015
00016 /*******************************************************************************
00017  * INCLUDE HEADER FILES
00018  ******************************************************************************/
00019
00020 #include "queue.h"
00021
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

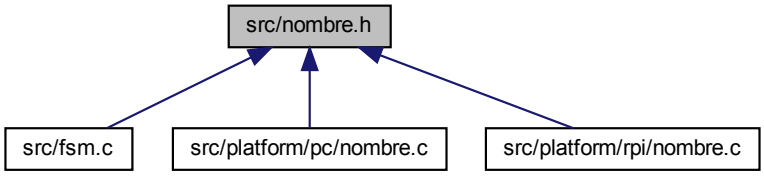
```
00001
00012 #ifndef _MENU_H_
00013 #define _MENU_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  ****************************************************************************/
00018
00019 #include "queue.h"
00020
00021 /*****************************************************************************
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

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

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_TUTORIAL "sprite_tutorial2"
- #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_mute (void)

  Mutea completamente el sonido de allegro.
- void allegro_sound_unmute (void)

  Desmutea el sonido de allegro.
- 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 72 of file allegro_stuff.c.

4.16.2.2 EXTENSION_SOUND_STREAM `#define EXTENSION_SOUND_STREAM ".opus"`

Definition at line 73 of file allegro_stuff.c.

4.16.2.3 EXTENSION_SPRITES `#define EXTENSION_SPRITES ".png"`

Definition at line 74 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 83 of file allegro_stuff.c.

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

Definition at line 79 of file allegro_stuff.c.

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

Definition at line 81 of file allegro_stuff.c.

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

Definition at line 78 of file allegro_stuff.c.

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

Definition at line 77 of file allegro_stuff.c.

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

Definition at line 80 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 67 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 69 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 66 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 65 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 68 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.

4.16.2.40 SPRITE_TUTORIAL `#define SPRITE_TUTORIAL "sprite_tutorial2"`

Definition at line 64 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 139 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 482 of file allegro_stuff.c.

4.16.4.2 allegro_deinit_display() `void allegro_deinit_display (`

```
void )
```

Desinicializa el display.

Definition at line 400 of file allegro_stuff.c.

4.16.4.3 allegro_deinits() `void allegro_deinits (`

```
void )
```

Desinicializaciones de allegro.

Definition at line 343 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 487 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 777 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 492 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 502 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 415 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 432 of file allegro_stuff.c.

4.16.4.10 allegro_get_rick_flag() `bool allegro_get_rick_flag ( void )`

Definition at line 795 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 447 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 442 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 467 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 472 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 477 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 452 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 292 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 497 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 359 of file allegro_stuff.c.

4.16.4.20 allegro_rick_draw() void allegro_rick_draw (void)

Definition at line 812 of file allegro_stuff.c.

4.16.4.21 allegro_rick_off() void allegro_rick_off (void)

Definition at line 805 of file allegro_stuff.c.

4.16.4.22 allegro_rick_on() void allegro_rick_on (void)

Definition at line 787 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 420 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 800 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 457 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 507 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 462 of file allegro_stuff.c.

4.16.4.28 allegro_sound_mute() void allegro_sound_mute (void)

Mutea completamente el sonido de allegro.

Definition at line 692 of file allegro_stuff.c.

4.16.4.29 allegro_sound_pause_stream() `void allegro_sound_pause_stream ( void )`

Pausa la musica actual, si hay alguna seleccionada.

Definition at line 655 of file allegro_stuff.c.

4.16.4.30 allegro_sound_play_effect_bonus() `void allegro_sound_play_effect_bonus ( void )`

Reproduce efecto de bonus.

Definition at line 708 of file allegro_stuff.c.

4.16.4.31 allegro_sound_play_effect_click() `void allegro_sound_play_effect_click ( void )`

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

Definition at line 713 of file allegro_stuff.c.

4.16.4.32 allegro_sound_play_effect_coin_drop() `void allegro_sound_play_effect_coin_drop ( void )`

Reproduce efecto `de moneda tirada`

Definition at line 768 of file allegro_stuff.c.

4.16.4.33 allegro_sound_play_effect_crash() `void allegro_sound_play_effect_crash ( void )`

Reproduce efecto de choque.

Definition at line 718 of file allegro_stuff.c.

4.16.4.34 allegro_sound_play_effect_drowned() `void allegro_sound_play_effect_drowned ( void )`

Reproduce efecto de ahogado (toco agua)

Definition at line 723 of file allegro_stuff.c.

4.16.4.35 allegro_sound_play_effect_exiting() `void allegro_sound_play_effect_exiting ( void )`

Reproduce efecto de 'saliendo'.

Definition at line 758 of file allegro_stuff.c.

4.16.4.36 allegro_sound_play_effect_goal() `void allegro_sound_play_effect_goal ( void )`

Reproduce efecto de 'llego a la meta'.

Definition at line 728 of file allegro_stuff.c.

4.16.4.37 allegro_sound_play_effect_jump() `void allegro_sound_play_effect_jump ( void )`

Reproduce efecto de salto.

Definition at line 733 of file allegro_stuff.c.

4.16.4.38 allegro_sound_play_effect_low_time() `void allegro_sound_play_effect_low_time ( void )`

Reproduce efecto de 'queda poco tiempo'.

Definition at line 738 of file allegro_stuff.c.

4.16.4.39 allegro_sound_play_effect_menu_enter() `void allegro_sound_play_effect_menu_enter ( void )`

Reproduce efecto de 'menu enter'.

Definition at line 748 of file allegro_stuff.c.

4.16.4.40 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 753 of file allegro_stuff.c.

4.16.4.41 allegro_sound_play_effect_no_time() `void allegro_sound_play_effect_no_time ( void )`

Reproduce efecto de 'sin tiempo'.

Definition at line 763 of file allegro_stuff.c.

4.16.4.42 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 743 of file allegro_stuff.c.

4.16.4.43 allegro_sound_play_stream() `void allegro_sound_play_stream ( void )`

Reproduce la musica actual, si hay alguna seleccionada.

Definition at line 646 of file allegro_stuff.c.

4.16.4.44 allegro_sound_restart_stream() `void allegro_sound_restart_stream ( void )`

Reinicia la musica actual, si hay alguna seleccionada.

Definition at line 664 of file allegro_stuff.c.

4.16.4.45 allegro_sound_set_stream_credits() `void allegro_sound_set_stream_credits ( void )`

Selecciona musica de creditos. Comienza pausada.

Definition at line 515 of file allegro_stuff.c.

4.16.4.46 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 682 of file allegro_stuff.c.

4.16.4.47 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 672 of file allegro_stuff.c.

4.16.4.48 allegro_sound_set_stream_game_over() `void allegro_sound_set_stream_game_over ( void )`

Selecciona musica de game over. Comienza pausada.

Definition at line 611 of file allegro_stuff.c.

4.16.4.49 allegro_sound_set_stream_main_menu() `void allegro_sound_set_stream_main_menu ( void )`

Selecciona musica de menu. Comienza pausada.

Definition at line 531 of file allegro_stuff.c.

4.16.4.50 allegro_sound_set_stream_pause_menu() `void allegro_sound_set_stream_pause_menu ( void )`

Selecciona musica de pausa. Comienza pausada.

Definition at line 547 of file allegro_stuff.c.

4.16.4.51 allegro_sound_set_stream_playing() `void allegro_sound_set_stream_playing ( void )`

Selecciona musica de jugando. Comienza pausada.

Definition at line 579 of file allegro_stuff.c.

4.16.4.52 allegro_sound_set_stream_ranking() `void allegro_sound_set_stream_ranking ( void )`

Selecciona musica de ranking. Comienza pausada.

Definition at line 563 of file allegro_stuff.c.

4.16.4.53 allegro_sound_set_stream_rick() `void allegro_sound_set_stream_rick (`

```
void )
```

Definition at line 595 of file allegro_stuff.c.

4.16.4.54 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 630 of file allegro_stuff.c.

4.16.4.55 allegro_sound_unmute() `void allegro_sound_unmute (`

```
void )
```

Desmutea el sonido de allegro.

Definition at line 699 of file allegro_stuff.c.

4.16.4.56 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 425 of file allegro_stuff.c.

4.16.4.57 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 282 of file allegro_stuff.c.

4.16.5 Variable Documentation

4.16.5.1 sprites `sprites_t sprites`

Definition at line 152 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_TUTORIAL "sprite_tutorial2"
00065 #define SPRITE_ICON "icon"
00066 #define SPRITE_DEAD "sprite_dead"
00067 #define SPRITE_BORDER "sprite_border"
00068 #define SPRITE_SPLASH "sprite_splash"
00069 #define SPRITE_COIN "sprite_coin"
00070
00071 // Extensiones
00072 #define EXTENSION_SOUND_SAMPLE ".wav"
00073 #define EXTENSION_SOUND_STREAM ".opus"
00074 #define EXTENSION_SPRITES ".png"
00075
00076 // Local paths
00077 #define PATH_SOUND_STREAMS "../res/sounds/streams/"
```

```
00078 #define PATH_SOUND_SAMPLES "../res/sounds/samples/"
00079 #define PATH_FONTS "../res/fonts/"
00080 #define PATH_SPRITES "../res/sprites/"
00081 #define PATH_GIFS "../res/gifs/"
00082
00083 #define GLOBAL_STREAM_VOLUME (double)0.5
00084
00085 /*****************************************************************************
00086  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00087  *****************************************************************************/
00088
00089 typedef struct
00090 {
00092     ALLEGRO_TIMER *timer;
00093
00094     // cola de eventos
00095     ALLEGRO_EVENT_QUEUE *queue;
00096
00097     // display
00098     ALLEGRO_DISPLAY *disp;
00099
00100     // fuente builtin
00101     ALLEGRO_FONT *font;
00102     int font_h; // altura de un caracter
00103     int font_w; // ancho de un caracter
00104
00105     // variable evento
00106     ALLEGRO_EVENT event;
00107
00108     // flag para salir el programa
00109     bool done;
00110     // flag para renderizar
00111     bool redraw;
00112
00113 } allegro_t;
00114
00115 typedef struct
00116 {
00117     ALLEGRO_AUDIO_STREAM *stream;
00118     unsigned char stream_state;
00119
00120     struct
00121     {
00122         ALLEGRO_SAMPLE *jump;
00123         ALLEGRO_SAMPLE *crash;
00124         ALLEGRO_SAMPLE *goal;
00125         ALLEGRO_SAMPLE *low_time;
00126         ALLEGRO_SAMPLE *click;
00127         ALLEGRO_SAMPLE *bonus;
00128         ALLEGRO_SAMPLE *run_completed;
00129         ALLEGRO_SAMPLE *drowned;
00130         ALLEGRO_SAMPLE *menu_enter;
00131         ALLEGRO_SAMPLE *new_max_score;
00132         ALLEGRO_SAMPLE *exiting;
00133         ALLEGRO_SAMPLE *no_time;
00134         ALLEGRO_SAMPLE *coin_drop;
00135     } samples;
00136
00137 } sounds_t;
00138
00139 enum SOUND_STREAM_STATES
00140 {
00141     SOUND_STREAM_STATE_NO_INIT,
00142     SOUND_STREAM_STATE_INIT,
00143     SOUND_STREAM_STATE_PAUSE,
00144     SOUND_STREAM_STATE_PLAY
00145 };
00146
00147 /*****************************************************************************
00148  * VARIABLES WITH GLOBAL SCOPE
00149  *****************************************************************************/
00150
00151 // estructura con punteros a sprites
00152 sprites_t sprites;
00153
00154 /*****************************************************************************
00155  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00156  *****************************************************************************/
00157
00169 static ALLEGRO_BITMAP *sprite_cut(ALLEGRO_BITMAP *source_bmp, int x, int y, int w, int h);
00170
00175 static void sprites_init(void);
00176
00192 static char *make_sprite_path(char *file_name, char *prev_str);
00193
00198 static void sprites_deinit(void);
00199
```

```
00204 static void audio_init(void);
00205
00210 static void audio_deinit(void);
00211
00220 static bool init_audio_stream(const char *file, float gain);
00221
00230 static bool init_sample(ALLEGRO_SAMPLE **sample, const char *file);
00231
00236 static void rick_init(void);
00237
00242 static void rick_deinit(void);
00243
00244 /*****************************************************************************
00245  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00246  *****************************************************************************/
00247
00248 // variables principales de allegro
00249 static allegro_t allegro_vars;
00250
00251 // Ultima tecla presionada
00252 static unsigned char last_key;
00253
00254 // variable con los sonidos/musicas del juego
00255 static sounds_t sounds;
00256
00257 // nombre del ultimo stream inicializado
00258 static char last_init_stream[30];
00259
00260 static ALGIF_ANIMATION *rick;
00261
00262 static char rick_prev_stream[30];
00263
00264 static bool rick_flag;
00265
00266 static ALLEGRO_MONITOR_INFO monitor_info;
00267
00268 static double stream_gain = GLOBAL_STREAM_VOLUME; //ganancia para musica y efectos
00269 static double last_stream_gain = GLOBAL_STREAM_VOLUME; //ultima ganancia antes del mute
00270
00271 static bool display_state = false;
00272
00273 //Para retener ultima posicion de ventana antes de recrearla
00274 static int window_x, window_y;
00275
00276 /*****************************************************************************
00277  *****************************************************************************
00278             GLOBAL FUNCTION DEFINITIONS
00279  *****************************************************************************
00280  *****************************************************************************/
00281
00282 void must_init(bool test, const char *description)
00283 {
00284     if (!test)
00285     {
00286         printf("~no se pudo inicializar %s~\n", description);
00287         while (true)
00288             ;
00289     }
00290 }
00291
00292 void allegro_inits(void)
00293 {
00294     must_init(al_init(), "allegro");
00295     must_init(al_install_keyboard(), "keyboard");
00296     must_init(al_install_mouse(), "mouse");
00297     must_init(al_init_image_addon(), "image");
00298     al_init_font_addon();
00299     must_init(al_init_ttf_addon(), "ttf addon");
00300
00301     // timer que actualiza cada 1/60 segundos (60fps)
00302     allegro_vars.timer = al_create_timer(1.0 / FPS);
00303     must_init(allegro_vars.timer, "timer");
00304
00305     // cola de eventos
00306     allegro_vars.queue = al_create_event_queue();
00307     must_init(allegro_vars.queue, "queue");
00308
00309     // Inicializa los spritesheets.
00310     sprites_init();
00311
00312     // para dibujar figuras primitivas (círculos, rectángulos, líneas, rellenos o no, etc.)
00313     must_init(al_init_primitives_addon(), "primitives");
00314
00315     // registra eventos posibles
00316     al_register_event_source(allegro_vars.queue, al_get_keyboard_event_source());
00317     al_register_event_source(allegro_vars.queue, al_get_timer_event_source(allegro_vars.timer));
00318     al_register_event_source(allegro_vars.queue, al_get_mouse_event_source());
```

```
00319
00320     // flag para salir el programa
00321     allegro_vars.done = false;
00322     // flag para renderizar
00323     allegro_vars.redraw = false;
00324
00325     // audio
00326     must_init(al_install_audio(), "audio");
00327     must_init(al_init_acodec_addon(), "audio codecs");
00328     must_init(al_reserve_samples(16), "reserve samples");
00329
00330     audio_init();
00331
00332     rick_init();
00333
00334     must_init(al_get_monitor_info(0, &monitor_info), "getting monitor info");
00335
00336     // creacion del display
00337     allegro_reinit_display();
00338
00339     // inicializa timer
00340     al_start_timer(allegro_vars.timer);
00341 }
00342
00343 void allegro_deinits(void)
00344 {
00345     rick_deinit();
00346     sprites_deinit();
00347     audio_deinit();
00348
00349     if(display_state)
00350     {
00351         al_destroy_font(allegro_vars.font);
00352         al_destroy_display(allegro_vars.disp);
00353     }
00354
00355     al_destroy_timer(allegro_vars.timer);
00356     al_destroy_event_queue(allegro_vars.queue);
00357 }
00358
00359 void allegro_reinit_display(void)
00360 {
00361
00362
00363     al_set_new_bitmap_flags(ALLEGRO_MIN_LINEAR | ALLEGRO_MAG_LINEAR);
00364     // Para tener aceleracion por HW desde la GPU (hace que no explote con los draw_text)
00365     al_set_new_bitmap_flags(ALLEGRO_VIDEO_BITMAP);
00366     // al_set_new_display_flags(ALLEGRO_RESIZABLE);
00367
00368     // Titulo de la ventana
00369     al_set_new_window_title("~ Programación I ~ TP Final ~ Frogger ~");
00370
00371     if(!window_x && !window_y)
00372         // Centrado en pantalla, según el monitor
00373         al_set_new_window_position(monitor_info.x2 / 2 - DISPLAY_W / 2, monitor_info.y2 / 2 -
      DISPLAY_H / 2 - 50);
00374     else
00375         al_set_new_window_position(window_x - 5, window_y - 29);
00376
00377     // opciones para el display (antialiasing)
00378     al_set_new_display_option(ALLEGRO_SAMPLE_BUFFERS, 1, ALLEGRO_SUGGEST);
00379     al_set_new_display_option(ALLEGRO_SAMPLES, 8, ALLEGRO_SUGGEST);
00380
00381     // creación del display
00382     allegro_vars.disp = al_create_display(DISPLAY_W, DISPLAY_H);
00383     must_init(allegro_vars.disp, "display");
00384     al_register_event_source(allegro_vars.queue, al_get_display_event_source(allegro_vars.disp));
00385
00386     al_set_display_icon(allegro_vars.disp, sprites.icon);
00387
00388     // Reload de la fuente
00389     char string[60] = PATH_FONTS;
00390     strcat(string, FONT_FILE_NAME);
00391     allegro_vars.font = al_load_font(string, FONT_HEIGHT, 0);
00392     must_init(allegro_vars.font, "font");
00393     allegro_vars.font_h = al_get_font_line_height(allegro_vars.font);
00394     allegro_vars.font_w = al_get_text_width(allegro_vars.font, "a");
00395
00396
00397     display_state = true;
00398 }
00399
00400 void allegro_deinit_display(void)
00401 {
00402     if (display_state)
00403     {
00404         al_get_window_position(allegro_vars.disp, &window_x, &window_y);
```

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

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

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

```
00665 {
00666     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00667     {
00668         init_audio_stream(last_init_stream, stream_gain);
00669     }
00670 }
00671
00672 void allegro_sound_set_stream_gain_up(void)
00673 {
00674     if (stream_gain <= 0.9)
00675     {
00676         stream_gain += 0.1;
00677         last_stream_gain = stream_gain;
00678         al_set_audio_stream_gain(sounds.stream, stream_gain);
00679     }
00680 }
00681
00682 void allegro_sound_set_stream_gain_down(void)
00683 {
00684     if (stream_gain >= 0.1)
00685     {
00686         stream_gain -= 0.1;
00687         last_stream_gain = stream_gain;
00688         al_set_audio_stream_gain(sounds.stream, stream_gain);
00689     }
00690 }
00691
00692 void allegro_sound_mute(void)
00693 {
00694     last_stream_gain = stream_gain;
00695     stream_gain = 0;
00696     al_set_audio_stream_gain(sounds.stream, stream_gain);
00697 }
00698
00699 void allegro_sound_unmute(void)
00700 {
00701     stream_gain = last_stream_gain;
00702     al_set_audio_stream_gain(sounds.stream, stream_gain);
00703 }
00704
00705 #pragma endregion allegro_sound_control
00706
00707 #pragma region allegro_sound_play_sample
00708 void allegro_sound_play_effect_bonus(void)
00709 {
00710     al_play_sample(sounds.samples.bonus, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00711 }
00712
00713 void allegro_sound_play_effect_click(void)
00714 {
00715     al_play_sample(sounds.samples.click, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00716 }
00717
00718 void allegro_sound_play_effect_crash(void)
00719 {
00720     al_play_sample(sounds.samples.crash, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00721 }
00722
00723 void allegro_sound_play_effect_drowned(void)
00724 {
00725     al_play_sample(sounds.samples.drowned, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00726 }
00727
00728 void allegro_sound_play_effect_goal(void)
00729 {
00730     al_play_sample(sounds.samples.goal, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00731 }
00732
00733 void allegro_sound_play_effect_jump(void)
00734 {
00735     al_play_sample(sounds.samples.jump, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00736 }
00737
00738 void allegro_sound_play_effect_low_time(void)
00739 {
00740     al_play_sample(sounds.samples.low_time, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00741 }
00742
00743 void allegro_sound_play_effect_run_completed(void)
00744 {
00745     al_play_sample(sounds.samples.run_completed, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00746 }
00747
00748 void allegro_sound_play_effect_menu_enter(void)
00749 {
00750     al_play_sample(sounds.samples.menu_enter, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00751 }
```

```
00752
00753 void allegro_sound_play_effect_new_max_score(void)
00754 {
00755     al_play_sample(sounds.samples.new_max_score, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00756 }
00757
00758 void allegro_sound_play_effect_exiting(void)
00759 {
00760     al_play_sample(sounds.samples.exiting, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00761 }
00762
00763 void allegro_sound_play_effect_no_time(void)
00764 {
00765     al_play_sample(sounds.samples.no_time, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00766 }
00767
00768 void allegro_sound_play_effect_coin_drop(void)
00769 {
00770     al_play_sample(sounds.samples.coin_drop, stream_gain, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00771 }
00772
00773 #pragma endregion allegro_sound_play_sample
00774
00775 #pragma endregion allegro_sound
00776
00777 void allegro_draw_hitbox(int x, int y, int w, int h)
00778 {
00779     al_draw_rectangle(x,
00780                       y,
00781                       x + w,
00782                       y + h,
00783                       al_map_rgb(100, 100, 100),
00784                       1); // grosor
00785 }
00786
00787 void allegro_rick_on(void)
00788 {
00789     strcpy(rick_prev_stream, last_init_stream);
00790
00791     allegro_sound_set_stream_rick();
00792     allegro_sound_play_stream();
00793 }
00794
00795 bool allegro_get_rick_flag(void)
00796 {
00797     return rick_flag;
00798 }
00799
00800 void allegro_set_rick_flag(bool state)
00801 {
00802     rick_flag = state;
00803 }
00804
00805 void allegro_rick_off(void)
00806 {
00807     must_init(init_audio_stream(rick_prev_stream, stream_gain),
00808               "retornando stream ~~ sacando rick");
00809     allegro_sound_play_stream();
00810 }
00811
00812 void allegro_rick_draw(void)
00813 {
00814     al_draw_bitmap(algif_get_bitmap(rick, al_get_time()), 100, DISPLAY_H / 2, 0);
00815 }
00816
00817 /*****************************************************************************
00818  *****************************************************************************
00819             LOCAL FUNCTION DEFINITIONS
00820  *****************************************************************************
00821  *****************************************************************************/
00822
00823 static ALLEGRO_BITMAP *sprite_cut(ALLEGRO_BITMAP *source_bmp, int x, int y, int w, int h)
00824 {
00825     ALLEGRO_BITMAP *sprite = al_create_sub_bitmap(source_bmp, x, y, w, h);
00826     must_init(sprite, "sprite cut");
00827     return sprite;
00828 }
00829
00830 static void sprites_init(void)
00831 {
00832     int i, offset;
00833     pair_xy_t temp_xy;
00834     int temp_w, temp_h;
00835
00836     char *path = NULL;
00837
00838     // de la rana completo
```

```
00839     path = make_sprite_path(SPRITE_FROG, NULL);
00840     sprites.frog_uncut = al_load_bitmap(path);
00841
00842     // se particiona el de la rana en sus 8 partes
00843     for (i = 0; i < FROG_FRAMES; i++)
00844     {
00845         temp_xy = getXYFromFrogFrame(i);
00846
00847         if (!(i % 2)) // los sprites pares
00848         {
00849             temp_w = SPRITE_SIZE_FROG_STATIC_W;
00850             temp_h = SPRITE_SIZE_FROG_STATIC_H;
00851         }
00852         else if (i == 1 || i == 7)
00853         {
00854             temp_w = SPRITE_SIZE_FROG_DYNAMIC_SHORT;
00855             temp_h = SPRITE_SIZE_FROG_DYNAMIC_LONG;
00856         }
00857         else
00858         {
00859             temp_w = SPRITE_SIZE_FROG_DYNAMIC_LONG;
00860             temp_h = SPRITE_SIZE_FROG_DYNAMIC_SHORT;
00861         }
00862
00863         sprites.frog[i] = sprite_cut(sprites.frog_uncut, temp_xy.x, temp_xy.y, temp_w, temp_h);
00864     }
00865
00866     // el del fondo
00867     path = make_sprite_path(SPRITE_BACKGROUND, path);
00868     sprites.background = al_load_bitmap(path);
00869
00870     // el de los troncos
00871     path = make_sprite_path(SPRITE_LOG, path);
00872     sprites.log = al_load_bitmap(path);
00873
00874     // recorte de los autos.
00875
00876     path = make_sprite_path(SPRITE_CAR, path);
00877     sprites.cars_uncut = al_load_bitmap(path);
00878
00879     sprites.car[0] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(0).x, getXYFromCarFrame(0).y,
      CAR_W, CAR_H);
00880     sprites.car[1] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(1).x, getXYFromCarFrame(1).y,
      CAR_W, CAR_H);
00881     sprites.car[2] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(2).x, getXYFromCarFrame(2).y,
      CAR_W, CAR_H);
00882     sprites.car[3] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(3).x, getXYFromCarFrame(3).y,
      CAR_TRUCK_FIRE_W, CAR_H);
00883     sprites.car[4] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(4).x, getXYFromCarFrame(4).y,
      CAR_TRUCK_W, CAR_H);
00884
00885     // el de las tortugas sin recortar
00886     path = make_sprite_path(SPRITE_TURTLES, path);
00887     sprites.turtle_uncut = al_load_bitmap(path);
00888
00889     // se recortan los de la tortuga en sus 11 partes
00890     for (i = 0; i < TURTLE_FRAMES; i++)
00891     {
00892         temp_xy = getXYFromTurtleFrame(i);
00893
00894         sprites.turtle[i] = sprite_cut(sprites.turtle_uncut, temp_xy.x, temp_xy.y, TURTLE_SIDE,
      TURTLE_SIDE);
00895     }
00896
00897     // corazon
00898     path = make_sprite_path(SPRITE_HEART, path);
00899     sprites.heart = al_load_bitmap(path);
00900
00901     // fondo de menu
00902     path = make_sprite_path(SPRITE_MENU_HOME_BACK, path);
00903     sprites.menu[MENU_WINDOW_HOME].background = al_load_bitmap(path);
00904
00905     // botones con highlight, sin recortar
00906     path = make_sprite_path(SPRITE_MENU_HOME, path);
00907     sprites.menu[MENU_WINDOW_HOME].uncut = al_load_bitmap(path);
00908
00909     // se recortan los highlight
00910     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00911     {
00912         sprites.menu[MENU_WINDOW_HOME].option[i] = sprite_cut(sprites.menu[MENU_WINDOW_HOME].uncut,
00913                                                               MENU_OPTION_TOPLEFT_X,
00914                                                               MENU_OPTION_TOPLEFT_Y + offset,
00915                                                               MENU_OPTION_W,
00916                                                               MENU_OPTION_H);
00917     }
00918
00919     path = make_sprite_path(SPRITE_MENU_DIFF_BACK, path);
```

```
00920     sprites.menu[MENU_WINDOW_DIFFICULTY].background = al_load_bitmap(path);
00921
00922     path = make_sprite_path(SPRITE_MENU_DIFF, path);
00923     sprites.menu[MENU_WINDOW_DIFFICULTY].uncut = al_load_bitmap(path);
00924
00925     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00926     {
00927         sprites.menu[MENU_WINDOW_DIFFICULTY].option[i] =
      sprite_cut(sprites.menu[MENU_WINDOW_DIFFICULTY].uncut,
00928                                                                 MENU_OPTION_TOPLEFT_X,
00929                                                                 MENU_OPTION_TOPLEFT_Y + offset,
00930                                                                 MENU_OPTION_W,
00931                                                                 MENU_OPTION_H);
00932     }
00933
00934     path = make_sprite_path(SPRITE_MENU_PAUSE_BACK, path);
00935     sprites.menu[MENU_WINDOW_PAUSE].background = al_load_bitmap(path);
00936
00937     path = make_sprite_path(SPRITE_MENU_PAUSE, path);
00938     sprites.menu[MENU_WINDOW_PAUSE].uncut = al_load_bitmap(path);
00939
00940     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00941     {
00942         sprites.menu[MENU_WINDOW_PAUSE].option[i] = sprite_cut(sprites.menu[MENU_WINDOW_PAUSE].uncut,
00943                                                                MENU_OPTION_TOPLEFT_X,
00944                                                                MENU_OPTION_TOPLEFT_Y + offset,
00945                                                                MENU_OPTION_W,
00946                                                                MENU_OPTION_H);
00947     }
00948
00949     path = make_sprite_path(SPRITE_MENU_GAME_OVER_BACK, path);
00950     sprites.menu[MENU_WINDOW_GAME_OVER].background = al_load_bitmap(path);
00951
00952     path = make_sprite_path(SPRITE_MENU_GAME_OVER, path);
00953     sprites.menu[MENU_WINDOW_GAME_OVER].uncut = al_load_bitmap(path);
00954
00955     for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00956     {
00957         sprites.menu[MENU_WINDOW_GAME_OVER].option[i] =
      sprite_cut(sprites.menu[MENU_WINDOW_GAME_OVER].uncut,
00958                                                                   MENU_OPTION_TOPLEFT_X,
00959                                                                   MENU_OPTION_TOPLEFT_Y + offset,
00960                                                                   MENU_OPTION_W,
00961                                                                   MENU_OPTION_H);
00962     }
00963
00964     path = make_sprite_path(SPRITE_CREDITS, path);
00965     sprites.credits = al_load_bitmap(path);
00966
00967     path = make_sprite_path(SPRITE_NAME, path);
00968     sprites.name = al_load_bitmap(path);
00969
00970     path = make_sprite_path(SPRITE_TUTORIAL, path);
00971     sprites.tutorial = al_load_bitmap(path);
00972
00973     path = make_sprite_path(SPRITE_ICON, path);
00974     sprites.icon = al_load_bitmap(path);
00975
00976     path = make_sprite_path(SPRITE_DEAD, path);
00977     sprites.dead = al_load_bitmap(path);
00978
00979     path = make_sprite_path(SPRITE_COIN, path);
00980     sprites.coin.uncut = al_load_bitmap(path);
00981     for (i = 0; i < SPRITE_COIN_FRAMES; i++)
00982     {
00983         pair_xy_t coord = getXYFromCoinFrame(i);
00984         sprites.coin.frame[i] = sprite_cut(sprites.coin.uncut, coord.x, coord.y, SPRITE_COIN_SIDE,
      SPRITE_COIN_SIDE);
00985     }
00986
00987     path = make_sprite_path(SPRITE_SPLASH, path);
00988     sprites.splash.uncut = al_load_bitmap(path);
00989     for (i = 0; i < SPRITE_SPLASH_FRAMES; i++)
00990     {
00991         pair_xy_t coord = getXYFromSplashFrame(i);
00992         sprites.splash.frame[i] = sprite_cut(sprites.splash.uncut, coord.x, coord.y, SPRITE_SPLASH_W,
      SPRITE_SPLASH_H);
00993     }
00994
00995     path = make_sprite_path(SPRITE_BORDER, path);
00996     sprites.border = al_load_bitmap(path);
00997
00998     free(path);
00999 }
01000
01001 static char *make_sprite_path(char *file_name, char *prev_str)
01002 {
```

```
01003     if (prev_str != NULL)
01004         free(prev_str);
01005
01006     char *path = NULL;
01007
01008     int str_size = sizeof(PATH_SPRITES) + strlen(file_name) + sizeof(EXTENSION_SPRITES) + 1;
01009
01010     path = malloc(str_size);
01011     must_init(path, file_name);
01012
01013     memset(path, 0, str_size);
01014
01015     strcat(path, PATH_SPRITES);
01016     strcat(path, file_name);
01017     strcat(path, EXTENSION_SPRITES);
01018
01019     return path;
01020 }
01021
01022 static void sprites_deinit(void)
01023 {
01024     int i, j;
01025
01026     al_destroy_bitmap(sprites.frog_uncut);
01027
01028     for (i = 0; i < FROG_FRAMES; i++)
01029         al_destroy_bitmap(sprites.frog[i]);
01030
01031     al_destroy_bitmap(sprites.background);
01032
01033     al_destroy_bitmap(sprites.log);
01034
01035     for (i = 0; i < CAR_TYPE_N; i++)
01036         al_destroy_bitmap(sprites.car[i]);
01037
01038     al_destroy_bitmap(sprites.turtle_uncut);
01039
01040     for (i = 0; i < TURTLE_FRAMES; i++)
01041         al_destroy_bitmap(sprites.turtle[i]);
01042
01043     al_destroy_bitmap(sprites.heart);
01044
01045     for (i = 0; i < MENU_WINDOW_MAX; i++)
01046     {
01047         al_destroy_bitmap(sprites.menu[i].background);
01048         al_destroy_bitmap(sprites.menu[i].uncut);
01049
01050         for (j = 0; j < MENU_STATE_MAX; j++)
01051         {
01052             al_destroy_bitmap(sprites.menu[i].option[j]);
01053         }
01054     }
01055
01056     al_destroy_bitmap(sprites.credits);
01057
01058     al_destroy_bitmap(sprites.name);
01059
01060     al_destroy_bitmap(sprites.tutorial);
01061
01062     al_destroy_bitmap(sprites.icon);
01063
01064     al_destroy_bitmap(sprites.dead);
01065
01066     al_destroy_bitmap(sprites.coin.uncut);
01067     for (i = 0; i < SPRITE_COIN_FRAMES; i++)
01068         al_destroy_bitmap(sprites.coin.frame[i]);
01069
01070     al_destroy_bitmap(sprites.splash.uncut);
01071     for (i = 0; i < SPRITE_SPLASH_FRAMES; i++)
01072         al_destroy_bitmap(sprites.splash.frame[i]);
01073
01074     al_destroy_bitmap(sprites.border);
01075 }
01076
01077 static void audio_init(void)
01078 {
01079     // streams
01080     sounds.stream_state = SOUND_STREAM_STATE_NO_INIT;
01081
01082     // efectos de sonido
01083     must_init(init_sample(&sounds.samples.bonus, "bonus_alert"),
01084               "effect_bonus sample");
01085
01086     must_init(init_sample(&sounds.samples.click, "click"),
01087               "effect_click sample");
01088
01089     must_init(init_sample(&sounds.samples.crash, "crash"),
```

```
01090               "effect_crash sample");
01091
01092     must_init(init_sample(&sounds.samples.drowned, "fall_in_water"),
01093               "effect_drowned sample");
01094
01095     must_init(init_sample(&sounds.samples.goal, "goal_reached"),
01096               "effect_goal sample");
01097
01098     must_init(init_sample(&sounds.samples.jump, "jump_original"),
01099               "effect_jump sample");
01100
01101     must_init(init_sample(&sounds.samples.low_time, "low_time_PC"),
01102               "effect_low_time sample");
01103
01104     must_init(init_sample(&sounds.samples.run_completed, "run_completed"),
01105               "effect_run_completed sample");
01106
01107     must_init(init_sample(&sounds.samples.menu_enter, "menu_enter"),
01108               "effect_menu_enter sample");
01109
01110     must_init(init_sample(&sounds.samples.new_max_score, "new_max_score"),
01111               "effect_new_max_score sample");
01112
01113     must_init(init_sample(&sounds.samples.exiting, "saliendo"),
01114               "effect_saliendo sample");
01115
01116     must_init(init_sample(&sounds.samples.no_time, "no_time"),
01117               "effect_no_time sample");
01118
01119     must_init(init_sample(&sounds.samples.coin_drop, "coin_drop"),
01120               "effect_coin_drop sample");
01121 }
01122
01123 static void audio_deinit(void)
01124 {
01125     if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
01126         al_destroy_audio_stream(sounds.stream);
01127
01128     al_destroy_sample(sounds.samples.bonus);
01129     al_destroy_sample(sounds.samples.click);
01130     al_destroy_sample(sounds.samples.crash);
01131     al_destroy_sample(sounds.samples.drowned);
01132     al_destroy_sample(sounds.samples.goal);
01133     al_destroy_sample(sounds.samples.jump);
01134     al_destroy_sample(sounds.samples.low_time);
01135     al_destroy_sample(sounds.samples.run_completed);
01136     al_destroy_sample(sounds.samples.menu_enter);
01137     al_destroy_sample(sounds.samples.no_time);
01138     al_destroy_sample(sounds.samples.coin_drop);
01139 }
01140
01141 static bool init_audio_stream(const char *file, float gain)
01142 {
01143     if (file == NULL)
01144         return false;
01145
01146     ALLEGRO_AUDIO_STREAM **pt = &sounds.stream;
01147     unsigned char *state = &sounds.stream_state;
01148
01149     int str_size = sizeof(PATH_SOUND_STREAMS) + strlen(file) + sizeof(EXTENSION_SOUND_STREAM) + 1;
01150     char *path = NULL;
01151
01152     // analisis de reproduccion y carga de stream "para que no explote todo"
01153     switch (*state)
01154     {
01155     case SOUND_STREAM_STATE_PLAY:
01156         // pausa
01157         al_set_audio_stream_playing(*pt, false);
01158
01159     case SOUND_STREAM_STATE_PAUSE:
01160         // desacople del mixer
01161         al_detach_audio_stream(*pt);
01162         // destruccion
01163         al_destroy_audio_stream(*pt);
01164
01165     case SOUND_STREAM_STATE_NO_INIT:
01166         // armado del path del archivo
01167         path = malloc(str_size);
01168         must_init(path, file);
01169         memset(path, 0, str_size);
01170         strcat(path, PATH_SOUND_STREAMS);
01171         strcat(path, file);
01172         strcat(path, EXTENSION_SOUND_STREAM);
01173
01174         // carga del stream
01175         *pt = al_load_audio_stream(path, 2, 2048);
01176         if (*pt == NULL)
```

```
01177             return false;
01178
01179         free(path);
01180
01181         // modo de reproduccion
01182         al_set_audio_stream_playmode(*pt, ALLEGRO_PLAYMODE_LOOP);
01183
01184         // ganancia
01185         al_set_audio_stream_gain(*pt, gain);
01186
01187         // pausa
01188         al_set_audio_stream_playing(*pt, false);
01189
01190         //"para que suene" (acople al mixer)
01191         al_attach_audio_stream_to_mixer(sounds.stream, al_get_default_mixer());
01192
01193         // actualiza el nombre del ultimo stream inicializado
01194         strcpy(last_init_stream, file);
01195
01196         *state = SOUND_STREAM_STATE_PAUSE;
01197         break;
01198
01199     default:
01200         break;
01201     }
01202
01203     return true;
01204 }
01205
01206 static bool init_sample(ALLEGRO_SAMPLE **sample, const char *file)
01207 {
01208     if (file == NULL)
01209         return false;
01210
01211     int str_size = sizeof(PATH_SOUND_SAMPLES) + strlen(file) + sizeof(EXTENSION_SOUND_SAMPLE) + 1;
01212     char *path = NULL;
01213
01214     path = malloc(str_size);
01215     must_init(path, file);
01216     memset(path, 0, str_size);
01217     strcat(path, PATH_SOUND_SAMPLES);
01218     strcat(path, file);
01219     strcat(path, EXTENSION_SOUND_SAMPLE);
01220
01221     *sample = al_load_sample(path);
01222
01223     free(path);
01224
01225     if (*sample == NULL)
01226         return false;
01227
01228     return true;
01229 }
01230
01231 static void rick_init(void)
01232 {
01233     rick = algif_load_animation("../res/gifs/rick.gif");
01234     allegro_set_rick_flag(false);
01235 }
01236
01237 static void rick_deinit()
01238 {
01239     algif_destroy_animation(rick);
01240 }
```

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_mute (void)

  Mutea completamente el sonido de allegro.
- void allegro_sound_unmute (void)

  Desmutea el sonido de allegro.
- 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 482 of file allegro_stuff.c.

4.18.4.2 allegro_deinit_display() `void allegro_deinit_display (`

```
void )
```

Desinicializa el display.

Definition at line 400 of file allegro_stuff.c.

4.18.4.3 allegro_deinits() `void allegro_deinits (`

```
void )
```

Desinicializaciones de allegro.

Definition at line 343 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 487 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 777 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 492 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 502 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 415 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 432 of file allegro_stuff.c.

4.18.4.10 allegro_get_rick_flag() `bool allegro_get_rick_flag ( void )`

Definition at line 795 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 447 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 442 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 467 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 472 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 477 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 452 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 292 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 497 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 359 of file allegro_stuff.c.

4.18.4.20 allegro_rick_draw() void allegro_rick_draw (void)

Definition at line 812 of file allegro_stuff.c.

4.18.4.21 allegro_rick_off() void allegro_rick_off (void)

Definition at line 805 of file allegro_stuff.c.

4.18.4.22 allegro_rick_on() void allegro_rick_on (void)

Definition at line 787 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 420 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 800 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 457 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 507 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 462 of file allegro_stuff.c.

4.18.4.28 allegro_sound_mute() void allegro_sound_mute (void)

Mutea completamente el sonido de allegro.

Definition at line 692 of file allegro_stuff.c.

4.18.4.29 allegro_sound_pause_stream() `void allegro_sound_pause_stream (void )`

Pausa la musica actual, si hay alguna seleccionada.

Definition at line 655 of file allegro_stuff.c.

4.18.4.30 allegro_sound_play_effect_bonus() `void allegro_sound_play_effect_bonus (void )`

Reproduce efecto de bonus.

Definition at line 708 of file allegro_stuff.c.

4.18.4.31 allegro_sound_play_effect_click() `void allegro_sound_play_effect_click (void )`

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

Definition at line 713 of file allegro_stuff.c.

4.18.4.32 allegro_sound_play_effect_coin_drop() `void allegro_sound_play_effect_coin_drop (void )`

Reproduce efecto `de moneda tirada`

Definition at line 768 of file allegro_stuff.c.

4.18.4.33 allegro_sound_play_effect_crash() `void allegro_sound_play_effect_crash (void )`

Reproduce efecto de choque.

Definition at line 718 of file allegro_stuff.c.

4.18.4.34 allegro_sound_play_effect_drowned() `void allegro_sound_play_effect_drowned (void )`

Reproduce efecto de ahogado (toco agua)

Definition at line 723 of file allegro_stuff.c.

4.18.4.35 allegro_sound_play_effect_exiting() `void allegro_sound_play_effect_exiting ( void )`

Reproduce efecto de 'saliendo'.

Definition at line 758 of file allegro_stuff.c.

4.18.4.36 allegro_sound_play_effect_goal() `void allegro_sound_play_effect_goal ( void )`

Reproduce efecto de 'llego a la meta'.

Definition at line 728 of file allegro_stuff.c.

4.18.4.37 allegro_sound_play_effect_jump() `void allegro_sound_play_effect_jump ( void )`

Reproduce efecto de salto.

Definition at line 733 of file allegro_stuff.c.

4.18.4.38 allegro_sound_play_effect_low_time() `void allegro_sound_play_effect_low_time ( void )`

Reproduce efecto de 'queda poco tiempo'.

Definition at line 738 of file allegro_stuff.c.

4.18.4.39 allegro_sound_play_effect_menu_enter() `void allegro_sound_play_effect_menu_enter ( void )`

Reproduce efecto de 'menu enter'.

Definition at line 748 of file allegro_stuff.c.

4.18.4.40 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 753 of file allegro_stuff.c.

4.18.4.41 allegro_sound_play_effect_no_time() `void allegro_sound_play_effect_no_time (void )`

Reproduce efecto de 'sin tiempo'.

Definition at line 763 of file allegro_stuff.c.

4.18.4.42 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 743 of file allegro_stuff.c.

4.18.4.43 allegro_sound_play_stream() `void allegro_sound_play_stream (void )`

Reproduce la musica actual, si hay alguna seleccionada.

Definition at line 646 of file allegro_stuff.c.

4.18.4.44 allegro_sound_restart_stream() `void allegro_sound_restart_stream (void )`

Reinicia la musica actual, si hay alguna seleccionada.

Definition at line 664 of file allegro_stuff.c.

4.18.4.45 allegro_sound_set_stream_credits() `void allegro_sound_set_stream_credits (void )`

Selecciona musica de creditos. Comienza pausada.

Definition at line 515 of file allegro_stuff.c.

4.18.4.46 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 682 of file allegro_stuff.c.

4.18.4.47 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 672 of file allegro_stuff.c.

4.18.4.48 allegro_sound_set_stream_game_over() `void allegro_sound_set_stream_game_over ( void )`

Selecciona musica de game over. Comienza pausada.

Definition at line 611 of file allegro_stuff.c.

4.18.4.49 allegro_sound_set_stream_main_menu() `void allegro_sound_set_stream_main_menu ( void )`

Selecciona musica de menu. Comienza pausada.

Definition at line 531 of file allegro_stuff.c.

4.18.4.50 allegro_sound_set_stream_pause_menu() `void allegro_sound_set_stream_pause_menu ( void )`

Selecciona musica de pausa. Comienza pausada.

Definition at line 547 of file allegro_stuff.c.

4.18.4.51 allegro_sound_set_stream_playing() `void allegro_sound_set_stream_playing ( void )`

Selecciona musica de jugando. Comienza pausada.

Definition at line 579 of file allegro_stuff.c.

4.18.4.52 allegro_sound_set_stream_ranking() `void allegro_sound_set_stream_ranking ( void )`

Selecciona musica de ranking. Comienza pausada.

Definition at line 563 of file allegro_stuff.c.

4.18.4.53 allegro_sound_set_stream_rick() `void allegro_sound_set_stream_rick (`

```
void )
```

Definition at line 595 of file allegro_stuff.c.

4.18.4.54 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 630 of file allegro_stuff.c.

4.18.4.55 allegro_sound_unmute() `void allegro_sound_unmute (`

```
void )
```

Desmutea el sonido de allegro.

Definition at line 699 of file allegro_stuff.c.

4.18.4.56 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 425 of file allegro_stuff.c.

4.18.4.57 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 282 of file allegro_stuff.c.

4.18.5 Variable Documentation

4.18.5.1 sprites `sprites_t sprites [extern]`

Definition at line 152 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 *tutorial;
00088
00089     ALLEGRO_BITMAP *icon;
00090
00091     ALLEGRO_BITMAP *dead;
00092
00093     struct
00094     {
00095         ALLEGRO_BITMAP *uncut;
00096         ALLEGRO_BITMAP *frame[SPRITE_COIN_FRAMES];
00097     } coin;
00098
00099     struct
00100     {
00101         ALLEGRO_BITMAP *uncut;
00102         ALLEGRO_BITMAP *frame[SPRITE_SPLASH_FRAMES];
00103     } splash;
00104
00105     ALLEGRO_BITMAP *border;
00106
00107 } sprites_t;
00108
00109 /*****************************************************************************
00110  * VARIABLE PROTOTYPES WITH GLOBAL SCOPE
00111  *****************************************************************************/
00112
00113 // estructura con punteros a sprites
00114 extern sprites_t sprites;
00115
00116 /*****************************************************************************
00117  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00118  *****************************************************************************/
00119
00126 void must_init(bool test, const char *description);
00127
00132 void allegro_inits(void);
00133
00138 void allegro_deinits(void);
00139
00144 void allegro_reinit_display(void);
00145
00150 void allegro_deinit_display(void);
00151
00157 unsigned char allegro_get_last_key(void);
00158
00164 void allegro_set_last_key(unsigned char allegro_key_code);
00165
00171 ALLEGRO_EVENT_TYPE allegro_wait_for_event(void);
00172
00178 ALLEGRO_EVENT *allegro_get_next_event(void);
00179
00185 ALLEGRO_EVENT allegro_get_var_event(void);
00186
00193 bool allegro_get_var_done(void);
00194
00201 bool allegro_get_var_redraw(void);
00202
00208 void allegro_set_var_done(bool state);
00209
00215 void allegro_set_var_redraw(bool state);
00216
00222 ALLEGRO_FONT *allegro_get_var_font(void);
00223
00229 int allegro_get_var_font_h(void);
00230
00236 int allegro_get_var_font_w(void);
00237
00242 void allegro_clear_display(void);
00243
00248 void allegro_draw_background(void);
00249
00255 void allegro_draw_menu_background(int window);
00256
00263 bool allegro_is_event_queueVacia(void);
00264
00269 ALLEGRO_EVENT_QUEUE *allegro_get_event_queue(void);
00270
00276 void allegro_set_var_event(ALLEGRO_EVENT event);
00277
00278 #pragma region allegro_sound
00279
```

```
00280 #pragma region allegro_sound_set_stream
00285 void allegro_sound_set_stream_credits(void);
00286
00291 void allegro_sound_set_stream_main_menu(void);
00292
00297 void allegro_sound_set_stream_pause_menu(void);
00298
00303 void allegro_sound_set_stream_ranking(void);
00304
00309 void allegro_sound_set_stream_playing(void);
00310
00315 void allegro_sound_set_stream_rick(void);
00316
00321 void allegro_sound_set_stream_game_over(void);
00322
00323 #pragma endregion allegro_sound_set_stream
00324
00325 #pragma region allegro_sound_control
00330 void allegro_sound_toggle_stream(void);
00331
00336 void allegro_sound_play_stream(void);
00337
00342 void allegro_sound_pause_stream(void);
00343
00348 void allegro_sound_restart_stream(void);
00349
00354 void allegro_sound_set_stream_gain_up(void);
00355
00360 void allegro_sound_set_stream_gain_down(void);
00361
00366 void allegro_sound_mute(void);
00367
00372 void allegro_sound_unmute(void);
00373
00374 #pragma endregion allegro_sound_control
00375
00376 #pragma region allegro_sound_play_sample
00381 void allegro_sound_play_effect_bonus(void);
00382
00387 void allegro_sound_play_effect_click(void);
00388
00393 void allegro_sound_play_effect_crash(void);
00394
00399 void allegro_sound_play_effect_drowned(void);
00400
00405 void allegro_sound_play_effect_goal(void);
00406
00411 void allegro_sound_play_effect_jump(void);
00412
00417 void allegro_sound_play_effect_low_time(void);
00418
00423 void allegro_sound_play_effect_run_completed(void);
00424
00429 void allegro_sound_play_effect_menu_enter(void);
00430
00435 void allegro_sound_play_effect_new_max_score(void);
00436
00441 void allegro_sound_play_effect_exiting(void);
00442
00447 void allegro_sound_play_effect_no_time(void);
00448
00453 void allegro_sound_play_effect_coin_drop(void);
00454
00455 #pragma endregion allegro_sound_play_sample
00456
00457 #pragma endregion allegro_sound
00458
00467 void allegro_draw_hitbox(int x, int y, int w, int h);
00468
00473 void allegro_rick_on(void);
00474
00479 bool allegro_get_rick_flag(void);
00480
00486 void allegro_set_rick_flag(bool state);
00487
00492 void allegro_rick_off(void);
00493
00498 void allegro_rick_draw(void);
00499
00500 /*****************************************************************************
00501  *****************************************************************************/
00502
00503 #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  * VARIABLES WITH GLOBAL SCOPE
00025  ******************************************************************************/
00026
00027 matriz_t disp_matriz;
00028
00029 /*******************************************************************************
00030  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00031  ******************************************************************************/
00032
00033 static void *threadTextoDisplay(void *ptr);
00034 static void *threadPresentacion(void *ptr);
00035 static void ulltoa(unsigned long long num, char *str);
00036 static int castear_posicion(int pos);
00037
00038 /*******************************************************************************
00039  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00040  ******************************************************************************/
00041
00042 static const clock_t TIEMPO_SLEEP_DISPLAY = CLOCKS_PER_SEC » 3;
00043
00044 /*******************************************************************************
00045  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00046  ******************************************************************************/
00047
00048 static pthread_mutex_t lock;
00049 static pthread_t ttextodisplay, tpresentacion;
00050 static mensaje_t texto1, texto2, texto3;
00051 static bool thread_encendido, thread_presentacion_encendido;
00052
00053 static int lines, i;
00054 static char **names;
00055 static unsigned long long *scores;
00056
00057 static char *autores[] = {"ALEJANDRO HEIR", "FRANCO AGGRIPINO", "MATIAS ALVAREZ", "TOMAS CASTRO"};
00058
00059 /*******************************************************************************
00060  *******************************************************************************
00061                         GLOBAL FUNCTION DEFINITIONS
00062  *******************************************************************************
00063  ******************************************************************************/
00064
00065 bool iniciarDisplay()
```

```
00066 {
00067     if (pthread_mutex_init(&lock, NULL) != 0)
00068     {
00069         return 1;
00070     }
00071 
00072     disp_init();  // inicializa el display
00073     disp_clear(); // limpia todo el display
00074 
00075     return 0;
00076 }
00077 
00078 void actualizarDisplay()
00079 {
00080     pthread_mutex_lock(&lock);
00081     for (int i = DISP_MIN; i <= (DISP_MAX_Y); i++)
00082         for (int j = DISP_MIN; j <= (DISP_MAX_X); j++)
00083             disp_write((dcoord_t){j, i}, disp_matriz[i] & (0x8000 » j));
00084 
00085     disp_update();
00086     pthread_mutex_unlock(&lock);
00087 }
00088 
00089 void limpiarDisplay()
00090 {
00091     texto1.habilitacion = false;
00092     texto2.habilitacion = false;
00093     texto3.habilitacion = false;
00094 
00095     if (thread_encendido)
00096     {
00097         thread_encendido = false;
00098         pthread_join(ttextodisplay, NULL);
00099     }
00100     if (thread_presentacion_encendido)
00101     {
00102         thread_presentacion_encendido = false;
00103         pthread_join(tpresentacion, NULL);
00104     }
00105 
00106     limpiarMatriz(disp_matriz);
00107     actualizarDisplay();
00108 }
00109 
00110 void mostrarTexto(char *txt, int pos)
00111 {
00112     int posicion = castear_posicion(pos);
00113     mensaje_t msj = mensaje(txt, posicion, false);
00114     while (!renglonIzquierdoLibre(&msj))
00115     {
00116         usleep(TIEMPO_SLEEP_DISPLAY);
00117         moverMensaje(&msj);
00118         copiarMatrizRenglon(disp_matriz, msj.renglon, msj.posicion);
00119         actualizarDisplay();
00120     }
00121 }
00122 
00123 void dejarTexto(char *txt, int pos, bool repetir)
00124 {
00125     int posicion = castear_posicion(pos);
00126 
00127     switch (posicion)
00128     {
00129     case POS_MSJ1:
00130         texto1 = mensaje(txt, posicion, repetir);
00131         break;
00132     case POS_MSJ2:
00133         texto2 = mensaje(txt, posicion, repetir);
00134         break;
00135     default:
00136         limpiarDisplay();
00137         texto3 = mensaje(txt, posicion, repetir);
00138     }
00139 
00140     if (!thread_encendido)
00141     {
00142         thread_encendido = true;
00143         pthread_create(&ttextodisplay, NULL, threadTextoDisplay, NULL);
00144     }
00145 }
00146 
00147 void cargarRanking(void)
00148 {
00149     borrarRenglon(texto2.renglon);
00150     lines = getRankingLineas();
00151     if (lines <= 0)
00152         dejarTexto("NINGUNA PARTIDA COMPLETADA AUN", POS_OPCION, true);
```

```
00153     else
00154     {
00155         names = getRankingNombres();
00156         scores = getRankingPuntos();
00157         i = 0;
00158     }
00159 }
00160
00161 void mostrarRanking(void)
00162 {
00163     if (renglonIzquierdoLibre(&texto2) && lines > 0)
00164     { // si se acabó lo que tenía para mostrar abajo, busco la siguiente posición
00165         renglon_t r = {0};
00166         uintARenglon(i + 1, r);
00167         copiarMatrizRenglon(disp_matriz, r, POS_MSJ1); // se pone la posición en el renglón de arriba
00168         actualizarDisplay();
00169
00170         char score_str[L_MAX], puesto_msj[L_MAX];
00171
00172         strcpy(puesto_msj, names[i]);
00173         strcat(puesto_msj, " ");
00174         ulltoa(scores[i], score_str);
00175         strcat(puesto_msj, score_str);           // se arma un string con el nombre de jugador y la
     puntuación
00176         dejarTexto(puesto_msj, POS_OPCION, false); // se muestra el string en la posición de abajo
     hasta que
00177
00178         if (++i >= lines) // apunto a la siguiente posición
00179             i = 0;
00180     }
00181 }
00182
00183 void cargarCreditos()
00184 {
00185     i = 0;
00186 }
00187
00188 void mostrarCreditos(void)
00189 {
00190     if (!texto3.habilitacion && !texto2.habilitacion && !thread_presentacion_encendido)
00191     {
00192         switch (i)
00193         {
00194     case 0:
00195       dejarTexto("PROGRAMACION TPF 2021 1C", POS_CREDITOS_INTRO, false);
00196       break;
00197         case 1:
00198             limpiarDisplay();
00199             thread_presentacion_encendido = true;
00200             pthread_create(&tpresentacion, NULL, threadPresentacion, NULL);
00201             break;
00202         case 2:
00203             pthread_join(tpresentacion, NULL);
00204       dejarTexto("AA", POS_MSJ_MENU, true);
00205         default:
00206             dejarTexto(autores[i-2], POS_OPCION, false);
00207         }
00208     if(++i > 5)
00209       i = 0;
00210     }
00211 }
00212
00213 void reconfigurarDisplayON(void)
00214 {
00215 }
00216
00217 void reconfigurarDisplayOFF(void)
00218 {
00219 }
00220
00221 /*******************************************************************************
00222  *******************************************************************************
00223                         LOCAL FUNCTION DEFINITIONS
00224  *******************************************************************************
00225  *******************************************************************************/
00226
00227 static void *threadTextoDisplay(void *ptr)
00228 {
00229     while (thread_encendido && (texto1.habilitacion || texto2.habilitacion || texto3.habilitacion))
00230     {
00231         usleep(TIEMPO_SLEEP_DISPLAY);
00232         if (texto1.habilitacion)
00233         {
00234             moverMensaje(&texto1);
00235             copiarMatrizRenglon(disp_matriz, texto1.renglon, POS_MSJ1);
00236         }
00237         if (texto2.habilitacion)
```

```
00238         {
00239             moverMensaje(&texto2);
00240             copiarMatrizRenglon(disp_matriz, texto2.renglon, POS_MSJ2);
00241         }
00242         if (texto3.habilitacion)
00243         {
00244             moverMensaje(&texto3);
00245             copiarMatrizRenglon(disp_matriz, texto3.renglon, POS_MSJ3);
00246         }
00247         actualizarDisplay();
00248     }
00249
00250     thread_encendido = false;
00251     pthread_exit(NULL);
00252 }
00253
00254 static void *threadPresentacion(void *ptr)
00255 {
00256     int coordenadas[][2] = {{0, 0}, {4, 1}, {8, 3}, {1, 8}, {5, 9}, {9, 10}, {13, 11}};
00257     int j;
00258   char str_presentacion[] = "FROGGER";
00259     for (j = 0; j < 14 && thread_presentacion_encendido; j++)
00260     {
00261         matriz_t letra_matriz;
00262         charAMatriz(str_presentacion[j % 7], letra_matriz, coordenadas[j % 7]);
00263         if (j >= 7)
00264             matrizXor(disp_matriz, letra_matriz);
00265         else
00266             matrizOr(disp_matriz, letra_matriz);
00267         actualizarDisplay();
00268         usleep(TIEMPO_SLEEP_DISPLAY);
00269     }
00270
00271     thread_presentacion_encendido = false;
00272
00273     return NULL;
00274 }
00275
00276 static void ulltoa(unsigned long long num, char *str)
00277 {
00278     unsigned long long sum = num;
00279     int i = 0;
00280     int digit;
00281     do
00282     {
00283         digit = sum % 10;
00284         str[i++] = '0' + digit;
00285         sum /= 10;
00286     } while (sum);
00287     str[i--] = '\0';
00288
00289     int j = 0;
00290     char ch;
00291     while (i > j)
00292     {
00293         ch = str[i];
00294         str[i--] = str[j];
00295         str[j++] = ch;
00296     }
00297 }
00298
00299 static int castear_posicion(int pos){
00300   if((pos == POS_CREDITOS_INTRO) || (pos == POS_MSJ_NEW_HI_SCORE))
00301     return POS_MSJ3;
00302   if((pos == POS_OPCION) || (pos == POS_CREDITOS))
00303     return POS_MSJ2;
00304   return POS_MSJ1;
00305 }
```

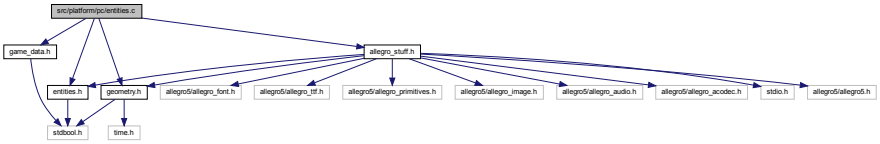
4.23 src/platform/pc/entities.c File Reference

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 8
- #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 CAR_SP_CH_WAIT FPS
- #define TURTLES_MIN_PER_PACK 1
- #define TURTLES_MAX_PER_PACK 3
- #define TURTLES_SPAWN_FRAMES 60
- #define TURTLES_SPAWN_MIN 1
- #define TURTLES_SPAWN_MAX 3
- #define TURTLES_MAX_USED 8
- #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 60
- #define TURTLES_SURFACE_FRAMES_MAX 600
- #define TURTLES_WATER_FRAMES_MIN 60
- #define TURTLES_WATER_FRAMES_MAX 200
- #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.
- void entities_set_tutorial (void)

  Setea el estado del flag tutorial.
- bool entities_get_tutorial (void)

  Indica el estado del flag tutorial.

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_SP_CH_WAIT `#define CAR_SP_CH_WAIT FPS`

Definition at line 41 of file entities.c.

4.23.2.2 CAR_SPEED_INCREASE `#define CAR_SPEED_INCREASE 2`

Definition at line 38 of file entities.c.

4.23.2.3 CAR_WAIT_INCREASE `#define CAR_WAIT_INCREASE 1`

Definition at line 39 of file entities.c.

4.23.2.4 CARS_BASE_SPEED `#define CARS_BASE_SPEED 1`

Definition at line 36 of file entities.c.

4.23.2.5 CARS_EXTRA_SEPARATOR `#define CARS_EXTRA_SEPARATOR CAR_W * 2`

Definition at line 40 of file entities.c.

4.23.2.6 CARS_MAX_USED `#define CARS_MAX_USED 15`

Definition at line 37 of file entities.c.

4.23.2.7 CARS_SPAWN_FRAMES `#define CARS_SPAWN_FRAMES 60`

Definition at line 35 of file entities.c.

4.23.2.8 CARS_SPAWN_MIN `#define CARS_SPAWN_MIN 2`

Definition at line 34 of file entities.c.

4.23.2.9 COIN_DESPAWN_FRAMES_MAX `#define COIN_DESPAWN_FRAMES_MAX 900`

Definition at line 63 of file entities.c.

4.23.2.10 COIN_DESPAWN_FRAMES_MIN `#define COIN_DESPAWN_FRAMES_MIN 700`

Definition at line 62 of file entities.c.

4.23.2.11 COIN_FRAME_RATE `#define COIN_FRAME_RATE 10`

Definition at line 64 of file entities.c.

4.23.2.12 COIN_FRAMES_TO_WARN_A `#define COIN_FRAMES_TO_WARN_A 250`

Definition at line 65 of file entities.c.

4.23.2.13 COIN_FRAMES_TO_WARN_B `#define COIN_FRAMES_TO_WARN_B 100`

Definition at line 66 of file entities.c.

4.23.2.14 COIN_SPAWN_FRAMES_MAX `#define COIN_SPAWN_FRAMES_MAX 600`

Definition at line 61 of file entities.c.

4.23.2.15 COIN_SPAWN_FRAMES_MIN `#define COIN_SPAWN_FRAMES_MIN 300`

Definition at line 60 of file entities.c.

4.23.2.16 COIN_WARNING_FRAMES_A `#define COIN_WARNING_FRAMES_A 20`

Definition at line 67 of file entities.c.

4.23.2.17 COIN_WARNING_FRAMES_B `#define COIN_WARNING_FRAMES_B 10`

Definition at line 68 of file entities.c.

4.23.2.18 LOGS_BASE_SPEED `#define LOGS_BASE_SPEED 1`

Definition at line 30 of file entities.c.

4.23.2.19 LOGS_EXTRA_SEPARATOR `#define LOGS_EXTRA_SEPARATOR LOG_W / 2`

Definition at line 32 of file entities.c.

4.23.2.20 LOGS_MAX_USED `#define LOGS_MAX_USED 8`

Definition at line 31 of file entities.c.

4.23.2.21 LOGS_SPAWN_FRAMES `#define LOGS_SPAWN_FRAMES 60`

Definition at line 29 of file entities.c.

4.23.2.22 LOGS_SPAWN_MAX `#define LOGS_SPAWN_MAX 3`

Definition at line 28 of file entities.c.

4.23.2.23 LOGS_SPAWN_MIN `#define LOGS_SPAWN_MIN 1`

Definition at line 27 of file entities.c.

4.23.2.24 SPRITE_DEAD_TIMEOUT `#define SPRITE_DEAD_TIMEOUT 80`

Definition at line 70 of file entities.c.

4.23.2.25 SPRITE_SPLASH_RATE `#define SPRITE_SPLASH_RATE 10`

Definition at line 72 of file entities.c.

4.23.2.26 TURTLES_BASE_SPEED `#define TURTLES_BASE_SPEED 2`

Definition at line 49 of file entities.c.

4.23.2.27 TURTLES_EXTRA_SEPARATOR `#define TURTLES_EXTRA_SEPARATOR TURTLE_SIDE * 2`

Definition at line 58 of file entities.c.

4.23.2.28 TURTLES_FRAME_TIMEOUT_GOING_DOWN `#define TURTLES_FRAME_TIMEOUT_GOING_DOWN 50`

Definition at line 51 of file entities.c.

4.23.2.29 TURTLES_FRAME_TIMEOUT_GOING_UP `#define TURTLES_FRAME_TIMEOUT_GOING_UP 10`

Definition at line 53 of file entities.c.

4.23.2.30 TURTLES_FRAME_TIMEOUT_SURFACE `#define TURTLES_FRAME_TIMEOUT_SURFACE 10`

Definition at line 50 of file entities.c.

4.23.2.31 TURTLES_FRAME_TIMEOUT_WATER `#define TURTLES_FRAME_TIMEOUT_WATER 20`

Definition at line 52 of file entities.c.

4.23.2.32 TURTLES_MAX_PER_PACK `#define TURTLES_MAX_PER_PACK 3`

Definition at line 44 of file entities.c.

4.23.2.33 TURTLES_MAX_USED `#define TURTLES_MAX_USED 8`

Definition at line 48 of file entities.c.

4.23.2.34 TURTLES_MIN_PER_PACK `#define TURTLES_MIN_PER_PACK 1`

Definition at line 43 of file entities.c.

4.23.2.35 TURTLES_SPAWN_FRAMES `#define TURTLES_SPAWN_FRAMES 60`

Definition at line 45 of file entities.c.

4.23.2.36 TURTLES_SPAWN_MAX `#define TURTLES_SPAWN_MAX 3`

Definition at line 47 of file entities.c.

4.23.2.37 TURTLES_SPAWN_MIN `#define TURTLES_SPAWN_MIN 1`

Definition at line 46 of file entities.c.

4.23.2.38 TURTLES_SURFACE_FRAMES_MAX `#define TURTLES_SURFACE_FRAMES_MAX 600`

Definition at line 55 of file entities.c.

4.23.2.39 TURTLES_SURFACE_FRAMES_MIN `#define TURTLES_SURFACE_FRAMES_MIN 60`

Definition at line 54 of file entities.c.

4.23.2.40 TURTLES_WATER_FRAMES_MAX `#define TURTLES_WATER_FRAMES_MAX 200`

Definition at line 57 of file entities.c.

4.23.2.41 TURTLES_WATER_FRAMES_MIN `#define TURTLES_WATER_FRAMES_MIN 60`

Definition at line 56 of file entities.c.

4.23.3 Enumeration Type Documentation

4.23.3.1 FROG_STATES `enum FROG_STATES`

Definition at line 157 of file entities.c.

4.23.3.2 TURTLE_STATES `enum TURTLE_STATES`

Definition at line 149 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 416 of file entities.c.

4.23.4.2 entities_get_tutorial() `bool entities_get_tutorial ( void )`

Indica el estado del flag tutorial.

Returns

true Tutorial done

false Playing tutorial

Definition at line 445 of file entities.c.

4.23.4.3 entities_init() `void entities_init (`

`void )`

Inicializa las entidades.

Definition at line 386 of file entities.c.

4.23.4.4 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 430 of file entities.c.

4.23.4.5 entities_set_tutorial() `void entities_set_tutorial ( void )`

Setea el estado del flag tutorial.

Definition at line 441 of file entities.c.

4.23.4.6 entities_update() `void entities_update ( void )`

Actualiza las entidades.

Definition at line 402 of file entities.c.

4.23.5 Variable Documentation

4.23.5.1 cont `unsigned int cont`

Definition at line 184 of file entities.c.

4.23.5.2 flag `bool flag`

Definition at line 173 of file entities.c.

4.23.5.3 frame_cont `unsigned int frame_cont`

Definition at line 183 of file entities.c.

4.23.5.4 timer `unsigned int timer`

Definition at line 174 of file entities.c.

4.23.5.5 x `unsigned int x`

Definition at line 175 of file entities.c.

4.23.5.6 y `unsigned int y`

Definition at line 176 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 8
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 #define CAR_SP_CH_WAIT FPS // frames para cambiar dfe velocidad
00042
00043 #define TURTLES_MIN_PER_PACK 1
00044 #define TURTLES_MAX_PER_PACK 3
00045 #define TURTLES_SPAWN_FRAMES 60 // cada cuantos frames spawnean
00046 #define TURTLES_SPAWN_MIN 1     // minimas a spawnear de una
00047 #define TURTLES_SPAWN_MAX 3     // maximas a spawnear de una
00048 #define TURTLES_MAX_USED 8      // maximas en pantalla
00049 #define TURTLES_BASE_SPEED 2
00050 #define TURTLES_FRAME_TIMEOUT_SURFACE 10    // cuanto "tiempo" dura un frame dibujado antes de pasar
       al siguiente
00051 #define TURTLES_FRAME_TIMEOUT_GOING_DOWN 50 // tiempo por frame al sumergirse
00052 #define TURTLES_FRAME_TIMEOUT_WATER 20      // tiempo por frame para mostrarse bajo el agua
00053 #define TURTLES_FRAME_TIMEOUT_GOING_UP 10   // tiempo por frame para mostrarse saliendo del agua
00054 #define TURTLES_SURFACE_FRAMES_MIN 60       // minimo "tiempo" en superficie
00055 #define TURTLES_SURFACE_FRAMES_MAX 600      // maximo "tiempo" en superficie
00056 #define TURTLES_WATER_FRAMES_MIN 60         // minimo "tiempo" bajo el agua
00057 #define TURTLES_WATER_FRAMES_MAX 200        // maximo "tiempo" bajo el agua
00058 #define TURTLES_EXTRA_SEPARATOR TURTLE_SIDE * 2
00059
00060 #define COIN_SPAWN_FRAMES_MIN 300   // mínimo tiempo para respawnear coin
00061 #define COIN_SPAWN_FRAMES_MAX 600   // maximo tiempo para respawnear coin
00062 #define COIN_DESPAWN_FRAMES_MIN 700 // mínimo tiempo para sacar coin
```

```
00063 #define COIN_DESPAWN_FRAMES_MAX 900 // maximo tiempo para sacar coin
00064 #define COIN_FRAME_RATE 10          // cada cuanto gira la coin
00065 #define COIN_FRAMES_TO_WARN_A 250   // frames previos al despawneo cuando empieza a titilar
00066 #define COIN_FRAMES_TO_WARN_B 100
00067 #define COIN_WARNING_FRAMES_A 20 // blink rate
00068 #define COIN_WARNING_FRAMES_B 10
00069
00070 #define SPRITE_DEAD_TIMEOUT 80 // frames que permanece en pantalla el sprite de muerte
00071
00072 #define SPRITE_SPLASH_RATE 10 // cada cuanto avanza un frame la animacion
00073
00074 /*******************************************************************************
00075  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00076  *******************************************************************************/
00077
00078 typedef struct
00079 {
00080     int x;
00081     int y;
00082     int moving;
00083     int facing;
00084     int steps;
00085     unsigned char state;
00086     unsigned char next_action;
00087
00088 } frog_t;
00089
00090 typedef struct
00091 {
00092     int x; // Posicion del auto
00093     int y;
00094     int lane;      // Carril del auto.
00095     int dx;        // Velocidad del auto.
00096     CAR_TYPE type; // Tipo de auto.
00097     int length;    // Largo del auto.
00098     int count;
00099     bool fast;
00100     bool used; // Marca disponibilidad en el array.
00101 } car_t;
00102
00103 typedef struct
00104 {
00105     int x;
00106     int y;
00107     int lane;
00108     int dx;
00109     bool used;
00110
00111 } log_t;
00112
00113 typedef struct
00114 {
00115     int x;                          // coordenada x
00116     int y;                          // coordenada y
00117     int lane;                       // carril
00118     int dx;                         // velocidad
00119     bool used;                      // flag de usada o no
00120     unsigned char turtles_in_pack; // cantidad de tortugas en el paquete
00121
00122     struct
00123     {
00124         unsigned char frame;  // contador que indica en qué frame de la animación se está (de 1 a
       TURTLES_FRAMES)
00125         unsigned int timeout; // timeout interno para cambiar de frame
00126         unsigned int cont;    // contador interno de frames de juego ejecutados
00127     } fx;
00128
00129     int wide;            // ancho del paquete, proporcional a turtles_in_pack y a TURTLES_SIDE
00130     unsigned char state; // estado (enum TURTLE_STATES)
00131 } turtle_pack_t;
00132
00133 typedef struct
00134 {
00135     int x;
00136     int y;
00137     bool used; // flag de usada o no
00138     struct
00139     {
00140         unsigned int frame_cont;  // contador de frame a mostrar
00141         unsigned int timeout;     // Para spawneo y despawneo
00142         unsigned int blink_timer; // Para titilar coin antes de sacarla
00143         unsigned int cont;        // contador interno de frames de juego ejecutados
00144         bool flag;                // Para indicar si debe parpadear o no
00145     } fx;
00146
00147 } coin_t;
00148
```

```
00149 enum TURTLE_STATES
00150 {
00151     TURTLE_STATE_SURFACE,
00152     TURTLE_STATE_GOING_DOWN,
00153     TURTLE_STATE_WATER,
00154     TURTLE_STATE_GOING_UP
00155 };
00156
00157 enum FROG_STATES
00158 {
00159     FROG_STATE_ROAD,
00160     FROG_STATE_WATER,
00161     FROG_STATE_LOG,
00162     FROG_STATE_TURTLE,
00163     FROG_STATE_GOAL,
00164     FROG_STATE_GOAL_COIN, // meta con coin
00165     FROG_STATE_CRASH_CAR,
00166     FROG_STATE_CRASH_WALL,
00167     FROG_STATE_BOUNCING_WALL // rebota contra algun borde
00168 };
00169
00170 // Estructur para administrar el sprite de muerte
00171 static struct
00172 {
00173     bool flag;          // para indicar graficar
00174     unsigned int timer; // contador para permanecer en pantalla
00175     unsigned int x;
00176     unsigned int y;
00177 } corpse_fx;
00178
00179 // Estructura para administrar el efecto de caida en agua
00180 static struct
00181 {
00182     bool flag;               // usado o no
00183     unsigned int frame_cont; // contador de frame a mostrar
00184     unsigned int cont;       // contador de ejecucion
00185     unsigned int x;          // X topleft
00186     unsigned int y;          // Y topleft
00187
00188 } splash_fx;
00189
00190 /**********************************************************************************
00191  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00192  *********************************************************************************/
00193
00198 static void frog_init(void);
00199
00204 static void frog_update(void);
00205
00210 static void frog_draw(void);
00211
00216 static void logs_init(void);
00217
00222 static void logs_update(void);
00223
00228 static void logs_draw(void);
00229
00234 static void cars_init(void);
00235
00240 static void cars_update(void);
00241
00246 static void cars_draw(void);
00247
00252 static void turtles_init(void);
00253
00258 static void turtles_update(void);
00259
00264 static void turtles_draw(void);
00265
00270 static void coin_init(void);
00271
00276 static void coin_update(void);
00277
00282 static void coin_draw(void);
00283
00288 // static void fix_frog_pos(void);
00289
00294 static void fix_frog_coord_y(void);
00295
00304 static bool is_frog_in_goal(void);
00305
00312 static void corpse_init(int x, int y);
00313
00318 static void corpse_update(void);
00319
00324 static void corpse_draw(void);
00325
```

```
00332 static void splash_init(int x, int y);
00333
00338 static void splash_update(void);
00339
00344 static void splash_draw(void);
00345
00346 /*********************************************************************************
00347  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00348  *********************************************************************************/
00349
00350 // Rana
00351 static frog_t frog;
00352
00353 // Array de troncos
00354 static log_t log[LOGS_MAX_USED];
00355
00356 // Array de autos
00357 static car_t car[CARS_MAX_USED];
00358
00359 // Array de paquetes de tortugas
00360 static turtle_pack_t turtle_pack[TURTLES_MAX_USED];
00361
00362 // Coin
00363 static coin_t coin;
00364
00365 // Contador de frames ejecutados
00366 static unsigned long game_frames;
00367
00368 // Carriles variables.
00369 static unsigned char normal_diff_lane;
00370 static unsigned char hard_diff_lane_1;
00371 static unsigned char hard_diff_lane_2;
00372
00373 // Maximo de autos spawneados.
00374
00375 static unsigned char cars_spawn_max;
00376
00377 // Flag muestra tutorial
00378
00379 static bool tutorial_flag;
00380 /*********************************************************************************
00381  *********************************************************************************
00382             GLOBAL FUNCTION DEFINITIONS
00383  *********************************************************************************
00384  *********************************************************************************/
00385
00386 void entities_init(void)
00387 {
00388     tutorial_flag = false;
00389     frog_init();
00390     logs_init();
00391     cars_init();
00392     turtles_init();
00393     coin_init();
00394
00395     game_frames = 0;
00396
00397     corpse_fx.flag = false;
00398
00399     splash_fx.flag = false;
00400 }
00401
00402 void entities_update()
00403 {
00404     game_frames = game_data_get_frames();
00405
00406     frog_update();
00407     logs_update();
00408     cars_update();
00409     turtles_update();
00410     coin_update();
00411
00412     corpse_update();
00413     splash_update();
00414 }
00415
00416 void entities_draw()
00417 {
00418     logs_draw();
00419     cars_draw();
00420     turtles_draw();
00421     coin_draw();
00422
00423     splash_draw();
00424     corpse_draw();
00425
00426     //"frog siempre a lo ultimo, para que se vea"
```

```
00427     frog_draw();
00428 }
00429
00430 void entities_move_frog(unsigned char direction)
00431 {
00432     if(tutorial_flag){
00433         if (direction == DIRECTION_DOWN || direction == DIRECTION_LEFT ||
00434             direction == DIRECTION_UP || direction == DIRECTION_RIGHT)
00435         {
00436             frog.next_action = direction;
00437         }
00438     }
00439 }
00440
00441 void entities_set_tutorial(void){
00442     tutorial_flag = true;
00443 }
00444
00445 bool entities_get_tutorial(void){
00446     return tutorial_flag;
00447 }
00448
00449
00450 /*****************************************************************************
00451  *****************************************************************************
00452             LOCAL FUNCTION DEFINITIONS
00453  *****************************************************************************
00454  *****************************************************************************/
00455
00456 static void frog_init(void)
00457 {
00458     frog.x = CELL_START_FROG_X;
00459     frog.y = CELL_START_FROG_Y;
00460     frog.moving = false;
00461     frog.facing = DIRECTION_UP;
00462     frog.steps = 0;
00463     frog.state = FROG_STATE_ROAD;
00464     frog.next_action = DIRECTION_NONE;
00465 }
00466
00467 static void frog_update(void)
00468 {
00469     int i;
00470
00471     bool interaction_flag = false;
00472
00473     if(!frog.moving)
00474     {
00475         if(frog.next_action == DIRECTION_DOWN || frog.next_action == DIRECTION_LEFT ||
00476             frog.next_action == DIRECTION_UP || frog.next_action == DIRECTION_RIGHT)
00477         {
00478             frog.facing = frog.next_action;
00479             frog.moving = true;
00480             frog.next_action = DIRECTION_NONE;
00481             allegro_sound_play_effect_jump();
00482         }
00483
00484     }
00485
00486     else if (frog.moving)
00487     {
00488
00489         if(frog.facing == DIRECTION_LEFT)
00490             frog.x -= STEP_FRACTION_SIZE;
00491         else if(frog.facing == DIRECTION_RIGHT)
00492             frog.x += STEP_FRACTION_SIZE;
00493         else if(frog.facing == DIRECTION_UP)
00494             frog.y -= STEP_FRACTION_SIZE;
00495         else if(frog.facing == DIRECTION_DOWN)
00496             frog.y += STEP_FRACTION_SIZE;
00497
00498         if(++frog.steps >= STEP_RATIO)
00499         {
00500             frog.steps = 0;
00501             frog.moving = false;
00502
00503             fix_frog_coord_y();
00504         }
00505     }
00506
00507
00508     //donde esta parada
00509     if(!frog.moving)
00510     {
00511         unsigned int y_no_offset = frog.y - FROG_OFFSET_Y;
00512
00513         //en alguna fila de descanso o de autos
```

```
00514         if(y_no_offset >= CELL_H * (lanes_cars[0] - 1) && y_no_offset <= DISPLAY_H - CELL_H)
00515             frog.state = FROG_STATE_ROAD;
00516 
00517         //en alguna fila de agua. Luego se actualiza si es sobre tronco o turtle
00518         else if(y_no_offset >= CELL_H * 2 && y_no_offset <= CELL_H * (lanes_cars[0] - 1))
00519             frog.state = FROG_STATE_WATER;
00520 
00521         //choque contra alguno de los muros superiores, o llegada bien a un goal
00522         else if (y_no_offset < CELL_H * 2)
00523         {
00524             if(!is_frog_in_goal())
00525             {
00526                 frog.state = FROG_STATE_CRASH_WALL;
00527             }
00528 
00529             else
00530             {
00531                 frog.state = FROG_STATE_GOAL;
00532 
00533                 //colision con coin
00534                 if(coin.used)
00535                 {
00536                     if(collideShort(    coin.x,
00537                                         coin.y,
00538                                         SPRITE_COIN_SIDE,
00539                                         SPRITE_COIN_SIDE,
00540                                         frog.x,
00541                                         frog.y,
00542                                         FROG_W,
00543                                         FROG_H))
00544                     {
00545                         frog.state = FROG_STATE_GOAL_COIN;
00546                         coin.used = false;
00547                     }
00548                 }
00549 
00550 
00551             }
00552 
00553             interaction_flag = true;
00554         }
00555 
00556         if(!interaction_flag)
00557         {
00558             //colision con autos
00559             for(i = 0; i < CARS_MAX_USED; i++)
00560             {
00561                 if(!car[i].used)
00562                     continue;
00563 
00564                 if(collideShort(    car[i].x,
00565                                     car[i].y,
00566                                     car[i].length,
00567                                     CAR_H,
00568                                     frog.x,
00569                                     frog.y,
00570                                     FROG_W,
00571                                     FROG_H))
00572                 {
00573                     frog.state = FROG_STATE_CRASH_CAR;
00574                     interaction_flag = true;
00575                     break;  //no puede chocar con 2 autos a la vez
00576                 }
00577             }
00578         }
00579 
00580     }
00581 
00582     if(!interaction_flag)
00583     {
00584         //esta en algun tronco?
00585         for(i = 0; i < LOGS_MAX_USED; i++)
00586         {
00587             if(!log[i].used)
00588                 continue;
00589 
00590             if(insideShortScaled(  log[i].x,
00591                                    log[i].y,
00592                                    LOG_W,
00593                                    LOG_H,
00594                                    frog.x,
00595                                    frog.y,
00596                                    FROG_W,
00597                                    FROG_H,
00598                                    INSERTION_FACTOR))
00599             {
00600                 frog.x += log[i].dx;
```

```
00601                 frog.state = FROG_STATE_LOG;
00602                 interaction_flag = true;
00603                 break;      //no puede estar en 2 troncos a la vez
00604             }
00605         }
00606     }
00607
00608     if(!interaction_flag)
00609     {
00610         //esta en algun turtle_pack?
00611         for(i = 0; i < TURTLES_MAX_USED; i++)
00612         {
00613             //Omite si el pack no esta usado o si esta bajo agua
00614             if(!turtle_pack[i].used || turtle_pack[i].state == TURTLE_STATE_WATER)
00615                 continue;
00616
00617             if(insideShortScaled(  turtle_pack[i].x,
00618                                    turtle_pack[i].y,
00619                                    turtle_pack[i].wide,
00620                                    TURTLE_SIDE,
00621                                    frog.x,
00622                                    frog.y,
00623                                    FROG_W,
00624                                    FROG_H,
00625                                    INSERTION_FACTOR))
00626             {
00627                 frog.x += turtle_pack[i].dx;
00628                 frog.state = FROG_STATE_TURTLE;
00629                 interaction_flag = true;
00630                 break;      //no puede estar en 2 packs a la vez
00631             }
00632         }
00633     }
00634
00635
00636     //revision de limites
00637     if(frog.x < FROG_MIN_X)
00638         frog.x = FROG_MIN_X;
00639     else if(frog.x > FROG_MAX_X)
00640         frog.x = FROG_MAX_X;
00641     else if(frog.y < FROG_MIN_Y)
00642         frog.y = FROG_MIN_Y;
00643     else if(frog.y > FROG_MAX_Y)
00644         frog.y = FROG_MAX_Y;
00645
00646     switch (frog.state)
00647     {
00648         case FROG_STATE_WATER:
00649             game_data_subtract_live();
00650             allegro_sound_play_effect_drowned();
00651
00652             splash_init(frog.x, frog.y);
00653
00654             frog_init();
00655
00656
00657             break;
00658
00659         case FROG_STATE_CRASH_CAR:
00660             game_data_subtract_live();
00661             allegro_sound_play_effect_crash();
00662
00663             corpse_init(frog.x, frog.y);
00664
00665             frog_init();
00666
00667             break;
00668
00669         case FROG_STATE_CRASH_WALL:
00670             game_data_subtract_live();
00671             allegro_sound_play_effect_crash();
00672
00673             corpse_init(frog.x, frog.y);
00674
00675             frog_init();
00676
00677             break;
00678
00679         case FROG_STATE_GOAL:
00680             game_data_add_run_time_goal();
00681             game_data_add_score();
00682             allegro_sound_play_effect_goal();
00683
00684             frog_init();
00685
00686             break;
00687
```

```
00688         case FROG_STATE_GOAL_COIN:
00689             game_data_add_run_time_goal_bonus();
00690             game_data_add_score_bonus();
00691             allegro_sound_play_effect_bonus();
00692 
00693             frog_init();
00694 
00695             break;
00696 
00697         default:
00698             break;
00699     }
00700 
00701 #ifdef DEBUG_ENTITIES_TEXT
00702     if(!(game_frames % 10))
00703         printf("state: %d ~~ y_no_offset: %d\n", frog.state, frog.y - FROG_OFFSET_Y);
00704 #endif
00705 
00706 }
00707 
00708 static void frog_draw(void)
00709 {
00710 
00711     ALLEGRO_BITMAP *tempbitmap = NULL;
00712 
00713     if (frog.moving)
00714     {
00715         if (frog.facing == DIRECTION_UP)
00716             tempbitmap = sprites.frog[1];
00717         if (frog.facing == DIRECTION_DOWN)
00718             tempbitmap = sprites.frog[7];
00719         if (frog.facing == DIRECTION_RIGHT)
00720             tempbitmap = sprites.frog[3];
00721         if (frog.facing == DIRECTION_LEFT)
00722             tempbitmap = sprites.frog[5];
00723     }
00724 
00725     else if (!frog.moving)
00726     {
00727         if (frog.facing == DIRECTION_UP)
00728             tempbitmap = sprites.frog[0];
00729         if (frog.facing == DIRECTION_DOWN)
00730             tempbitmap = sprites.frog[6];
00731         if (frog.facing == DIRECTION_RIGHT)
00732             tempbitmap = sprites.frog[2];
00733         if (frog.facing == DIRECTION_LEFT)
00734             tempbitmap = sprites.frog[4];
00735     }
00736 
00737     al_draw_bitmap(tempbitmap, frog.x, frog.y, 0);
00738 
00739 #ifdef DEBUG_ENTITIES_TEXT
00740     // hitbox
00741     allegro_draw_hitbox(frog.x, frog.y, FROG_W, FROG_H);
00742     // coordenadas rana
00743     al_draw_textf(allegro_get_var_font(), al_map_rgb(200, 50, 50), 0, 0, 0, "X: %d Y: %d", frog.x,
      frog.y);
00744 #endif
00745 }
00746 
00747 static void logs_init(void)
00748 {
00749     int i;
00750     for (i = 0; i < LOGS_MAX_USED; i++)
00751         log[i].used = false;
00752 }
00753 
00754 static void logs_update(void)
00755 {
00756     // se busca spawnear entre LOGS_SPAWN_MIN y LOGS_SPAWN_MAX autos cada LOGS_SPAWN_FRAMES frames
00757     int new_quota = ((game_frames % LOGS_SPAWN_FRAMES) ? 0 : get_rand_between(LOGS_SPAWN_MIN,
      LOGS_SPAWN_MAX));
00758 
00759     int i, used;
00760 
00761     // cuento cuantos troncos usados hay
00762     for (i = 0, used = 0; i < LOGS_MAX_USED; i++)
00763         used += log[i].used;
00764 
00765     for (i = 0; i < LOGS_MAX_USED; i++)
00766     {
00767         // Spawneo de troncos
00768         if (!log[i].used && new_quota > 0 && used < LOGS_MAX_USED) // Lugar libre?
00769         {
00770 
00771             // Asigno carril.
00772             int temp_rand_log_lane = get_rand_between(0, LANES_LOG_TOTAL - 1);
```

```
00773             log[i].lane = lanes_logs[temp_rand_log_lane];
00774
00775             // Coordenada 'y' en funcion del carril
00776             log[i].y = CELL_H * log[i].lane + LOG_OFFSET_Y;
00777
00778             // Velocidad
00779             // log[i].dx = lanes_logs[LANES_LOG_TOTAL-1] - log[i].lane + 1;
00780             // log[i].dx = map_int(log[i].lane, 0, lanes_logs[LANES_LOG_TOTAL-1], 1, 3);
00781             log[i].dx = log[i].lane - (temp_rand_log_lane + 2) + LOGS_BASE_SPEED;
00782
00783             // en pares...
00784             if (!(log[i].lane % 2))
00785             {
00786                 // coordenada de inicio
00787                 log[i].x = -LOG_W;
00788             }
00789
00790             // en impares...
00791             else
00792             {
00793                 // coordenada de inicio
00794                 log[i].x = DISPLAY_W;
00795
00796                 // hacia el otro lado
00797                 log[i].dx *= (-1);
00798             }
00799
00800             int p;
00801             bool check; // para confirmar asignacion de lane
00802             for (p = 0, check = true; p < LOGS_MAX_USED; p++)
00803             {
00804                 // si no es el mismo tronco, y ese otro esta usado, y coinciden en lane...
00805                 if (p != i && log[p].used && log[p].lane == log[i].lane)
00806                 {
00807                     // si colisiona con algun otro tronco...
00808                     if (collide(
00809                             log[i].x - LOGS_EXTRA_SEPARATOR,
00810                             log[i].y,
00811                             log[i].x + LOG_W + LOGS_EXTRA_SEPARATOR,
00812                             log[i].y + LOG_H,
00813                             log[p].x,
00814                             log[p].y,
00815                             log[p].x + LOG_W,
00816                             log[p].y + LOG_H))
00817                     {
00818                         // no spawnea
00819                         check = false;
00820                         break;
00821                     }
00822                 }
00823             }
00824
00825             // si se puede spawnear...
00826             if (check)
00827             {
00828                 // Pasa a usado
00829                 log[i].used = true;
00830                 used++;
00831                 new_quota--;
00832             }
00833
00834             // si no se puede spawnear...
00835             else
00836             {
00837             }
00838         }
00839
00840         // si el tronco esta usado...
00841         else if (log[i].used)
00842         {
00843             // desplaza
00844             log[i].x += log[i].dx;
00845
00846             // chequea si llego a los limites
00847             if ((log[i].dx > 0 && log[i].x >= DISPLAY_W) || (log[i].dx < 0 && log[i].x <= -LOG_W))
00848             {
00849                 log[i].used = false;
00850                 used--;
00851             }
00852
00853             // printf("~log%d lane%d dx%d~\n", i, log[i].lane, log[i].dx);
00854         }
00855     }
00856 }
00857
00858 static void logs_draw(void)
00859 {
```

```
00860     int i;
00861
00862     for (i = 0; i < LOGS_MAX_USED; i++)
00863     {
00864         if (log[i].used)
00865         {
00866             al_draw_bitmap(sprites.log, log[i].x, log[i].y, 0);
00867
00868 #ifdef DEBUG_ENTITIES_TEXT
00869             // hitbox
00870             allegro_draw_hitbox(log[i].x, log[i].y, LOG_W, LOG_H);
00871 #endif
00872         }
00873     }
00874
00875 #ifdef DEBUG_ENTITIES_TEXT
00876     // coordenadas
00877     int space;
00878     for (i = 0, space = 20; i < LOGS_MAX_USED; i++, space += 10)
00879     {
00880         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);
00881     }
00882 #endif
00883 }
00884
00885 static void cars_init(void)
00886 {
00887     int i;
00888     // Inicio array de autos desocupando.
00889     for (i = 0; i < CARS_MAX_USED; i++)
00890         car[i].used = false;
00891
00892     switch (game_data_get_diff())
00893     {
00894     case DIFFICULTIES_EASY:
00895         cars_spawn_max = 3;
00896         break;
00897
00898     case DIFFICULTIES_NORMAL:
00899         normal_diff_lane = get_rand_between(lanes_cars[0], lanes_cars[LANES_CAR_TOTAL - 1]);
00900         cars_spawn_max = 4;
00901         break;
00902
00903     case DIFFICULTIES_HARD:
00904         hard_diff_lane_1 = get_rand_between(lanes_cars[0], lanes_cars[2]);
00905         hard_diff_lane_2 = get_rand_between(lanes_cars[3], lanes_cars[4]);
00906         cars_spawn_max = 5;
00907         break;
00908     }
00909 }
00910
00911 static void cars_update(void)
00912 {
00913     // se busca spawnear entre CARS_SPAWN_MIN y cars_spawn_max autos cada CARS_SPAWN_FRAMES frames
00914     int new_quota = ((game_frames % CARS_SPAWN_FRAMES) ? 0 : get_rand_between(CARS_SPAWN_MIN,
       cars_spawn_max));
00915
00916     int i, used;
00917
00918     // cuento cuantos autos usados hay
00919     for (i = 0, used = 0; i < CARS_MAX_USED; i++)
00920         used += car[i].used;
00921
00922     for (i = 0; i < CARS_MAX_USED; i++)
00923     {
00924         // Spawneo de autos.
00925         if (!car[i].used && new_quota > 0 && used < CARS_MAX_USED) // Lugar libre?
00926         {
00927             // Asigno carril.
00928             car[i].lane = lanes_cars[get_rand_between(0, LANES_CAR_TOTAL - 1)];
00929
00930             // Coordenada 'y' en funcion del carril
00931             car[i].y = CELL_H * car[i].lane + CAR_OFFSET_Y;
00932
00933             // Velocidad menor en rutas mas alejadas
00934             car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED;
00935             // car[i].dx = CARS_BASE_SPEED;
00936
00937             // Asigno tipos.
00938             car[i].type = get_rand_between(0, CAR_TYPE_N - 1);
00939
00940             // Defino los largos correspondientes,
00941             switch (car[i].type)
00942             {
00943             case CAR_POLICE:
00944             case CAR_YELLOW:
```

```
00945             case CAR_BLUE:
00946                 car[i].length = CAR_W;
00947                 break;
00948             case TRUCK_FIRE:
00949                 car[i].length = CAR_TRUCK_FIRE_W;
00950                 break;
00951             case TRUCK:
00952                 car[i].length = CAR_TRUCK_W;
00953                 break;
00954             default:
00955                 break;
00956             }
00957 
00958             // Inicializo el contador;
00959             car[i].count = 0;
00960 
00961             // Inicializo el flag.
00962             car[i].fast = 0;
00963 
00964             // en pares...
00965             if (!(car[i].lane % 2))
00966             {
00967                 // coordenada de inicio
00968                 car[i].x = -car[i].length;
00969             }
00970 
00971             // en impares...
00972             else
00973             {
00974                 // coordenada de inicio
00975                 car[i].x = DISPLAY_W;
00976 
00977                 // hacia el otro lado
00978                 car[i].dx *= (-1);
00979             }
00980 
00981             int p;
00982             bool check; // para confirmar asignacion de lane
00983             for (p = 0, check = true; p < CARS_MAX_USED; p++)
00984             {
00985                 // si no es el mismo auto, y ese otro esta usado, y coinciden en lane...
00986                 if (p != i && car[p].used && car[p].lane == car[i].lane)
00987                 {
00988                     // si colisiona con algun otro auto...
00989                     if (collide(
00990                             car[i].x - CARS_EXTRA_SEPARATOR,
00991                             car[i].y,
00992                             car[i].x + car[i].length + CARS_EXTRA_SEPARATOR, // Es el mas largo.
00993                             car[i].y + CAR_H,
00994                             car[p].x,
00995                             car[p].y,
00996                             car[p].x + car[p].length,
00997                             car[p].y + CAR_H))
00998                     {
00999                         // no spawnea
01000                         check = false;
01001                         break;
01002                     }
01003                 }
01004             }
01005 
01006             // si se puede spawnear...
01007             if (check)
01008             {
01009                 // Pasa a usado
01010                 car[i].used = true;
01011                 used++;
01012                 new_quota--;
01013             }
01014 
01015             // si no se puede spawnear...
01016             else
01017             {
01018             }
01019         }
01020 
01021         // si el auto esta usado...
01022         else if (car[i].used)
01023         {
01024             // Carril con velocidad variable
01025             if (car[i].count < CAR_WAIT_INCREASE)
01026             {
01027                 switch (game_data_get_diff())
01028                 {
01029                 case DIFFICULTIES_EASY:
01030                     break;
01031 
```

```
01032                 case DIFFICULTIES_NORMAL:
01033                     if (car[i].lane == normal_diff_lane)
01034                     {
01035                         if (!(game_frames % CAR_SP_CH_WAIT))
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                     break;
01054                 case DIFFICULTIES_HARD:
01055                     if ((car[i].lane == hard_diff_lane_1) || (car[i].lane == hard_diff_lane_2))
01056                     {
01057                         if (!(game_frames % CAR_SP_CH_WAIT))
01058                         {
01059                             if (car[i].fast == 0)
01060                             {
01061                                 car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) +
      CARS_BASE_SPEED + CAR_SPEED_INCREASE;
01062                                 if (car[i].lane % 2)
01063                                     car[i].dx *= (-1);
01064                                 car[i].fast = 1;
01065                             }
01066                             else
01067                             {
01068                                 car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) +
      CARS_BASE_SPEED;
01069                                 if (car[i].lane % 2)
01070                                     car[i].dx *= (-1);
01071                                 car[i].fast = 0;
01072                             }
01073                         }
01074                     }
01075                 default:
01076                     break;
01077                 }
01078             }
01079             else
01080                 car[i].count++;
01081
01082             // Desplazamiento
01083             car[i].x += car[i].dx;
01084
01085             // chequea si llego a los limites
01086             if ((car[i].dx > 0 && car[i].x >= DISPLAY_W) || (car[i].dx < 0 && car[i].x <=
      -car[i].length))
01087             {
01088                 car[i].used = false;
01089                 used--;
01090             }
01091
01092             // printf("~car%d lane%d dx%d~\n", i, car[i].lane, car[i].dx);
01093         }
01094     }
01095 }
01096
01097 static void cars_draw()
01098 {
01099     int i;
01100     bool flag;
01101
01102     ALLEGRO_BITMAP *temp_bitmap = NULL;
01103
01104     for (i = 0; i < CARS_MAX_USED; i++)
01105     {
01106         if (car[i].used)
01107         {
01108             if (car[i].dx < 0)
01109                 flag = ALLEGRO_FLIP_HORIZONTAL;
01110             else
01111                 flag = 0;
01112
01113             temp_bitmap = sprites.car[car[i].type];
```

```
01114
01115             // Dibujo los autos en sus carriles.
01116             al_draw_bitmap(temp_bitmap, car[i].x, car[i].y, flag);
01117
01118 #ifdef DEBUG_ENTITIES_TEXT
01119             // Dibujo hitbox
01120             allegro_draw_hitbox(car[i].x, car[i].y, car[i].length, CAR_H);
01121 #endif
01122         }
01123     }
01124
01125 #ifdef DEBUG_ENTITIES_TEXT
01126     // coordenadas
01127     int space;
01128     for (i = 0, space = 200; i < CARS_MAX_USED; i++, space += 20)
01129     {
01130         // 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);
01131         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);
01132     }
01133 #endif
01134 }
01135
01136 static void turtles_init(void)
01137 {
01138     int i;
01139     for (i = 0; i < TURTLES_MAX_USED; i++)
01140     {
01141         turtle_pack[i].used = false;
01142     }
01143 }
01144
01145 static void turtles_update(void)
01146 {
01147     int new_quota = ((game_frames % TURTLES_SPAWN_FRAMES) ? 0 : get_rand_between(TURTLES_SPAWN_MIN,
       TURTLES_SPAWN_MAX));
01148
01149     int i, used;
01150
01151     for (i = 0, used = 0; i < TURTLES_MAX_USED; i++)
01152     {
01153         if (turtle_pack[i].used)
01154             used++;
01155     }
01156
01157     for (i = 0; i < TURTLES_MAX_USED; i++)
01158     {
01159         // Spawneo de turtle_packs
01160         if (!turtle_pack[i].used && new_quota > 0 && used < TURTLES_MAX_USED) // Lugar libre?
01161         {
01162
01163             // defino tortugas en el pack
01164             turtle_pack[i].turtles_in_pack = get_rand_between(TURTLES_MIN_PER_PACK,
       TURTLES_MAX_PER_PACK);
01165
01166             // calculo ancho del pack
01167             turtle_pack[i].wide = TURTLE_SIDE * turtle_pack[i].turtles_in_pack;
01168
01169             // Asigno carril.
01170             turtle_pack[i].lane = lanes_turtles[get_rand_between(0, LANES_TURTLE_TOTAL - 1)];
01171
01172             // Coordenada 'y' en funcion del carril
01173             turtle_pack[i].y = CELL_H * turtle_pack[i].lane;
01174
01175             // Velocidad
01176             // turtle_pack[i].dx = lanes_turtles[LANES_TURTLE_TOTAL- turtle_pack[i].lane + 1];
01177             turtle_pack[i].dx = TURTLES_BASE_SPEED;
01178
01179             // en pares...
01180             if (!(turtle_pack[i].lane % 2))
01181             {
01182                 // coordenada de inicio
01183                 turtle_pack[i].x = -turtle_pack[i].wide;
01184             }
01185
01186             // en impares...
01187             else
01188             {
01189                 // coordenada de inicio
01190                 turtle_pack[i].x = DISPLAY_W;
01191
01192                 // hacia el otro lado
01193                 turtle_pack[i].dx *= (-1);
01194             }
01195
01196             int p;
```

```
01197             bool check; // para confirmar asignacion de lane
01198             for (p = 0, check = true; p < TURTLES_MAX_USED; p++)
01199             {
01200                 // si no es el mismo pack, y ese otro esta usado, y coinciden en lane...
01201                 if (p != i && turtle_pack[p].used && turtle_pack[p].lane == turtle_pack[i].lane)
01202                 {
01203                     // si colisiona con algun otro pack...
01204                     if (collide(
01205                             turtle_pack[i].x - TURTLES_EXTRA_SEPARATOR,
01206                             turtle_pack[i].y,
01207                             turtle_pack[i].x + turtle_pack[i].wide + TURTLES_EXTRA_SEPARATOR,
01208                             turtle_pack[i].y + TURTLE_SIDE,
01209                             turtle_pack[p].x,
01210                             turtle_pack[p].y,
01211                             turtle_pack[p].x + turtle_pack[p].wide,
01212                             turtle_pack[p].y + TURTLE_SIDE))
01213                     {
01214                         // no spawnea
01215                         check = false;
01216                         break;
01217                     }
01218                 }
01219             }
01220 
01221             // si se puede spawnear...
01222             if (check)
01223             {
01224                 // Pasa a usado
01225                 turtle_pack[i].used = true;
01226                 used++;
01227 
01228                 // se inicializa el contador de frames
01229                 turtle_pack[i].fx.frame = 0;
01230                 turtle_pack[i].fx.cont = 1;
01231                 turtle_pack[i].fx.timeout = 0;
01232                 // fuera del agua
01233                 turtle_pack[i].state = TURTLE_STATE_SURFACE;
01234 
01235                 new_quota--;
01236             }
01237 
01238             // si no se puede spawnear...
01239             else
01240             {
01241             }
01242         }
01243 
01244         // si el pack esta usado...
01245         else if (turtle_pack[i].used)
01246         {
01247             // desplaza
01248             turtle_pack[i].x += turtle_pack[i].dx;
01249 
01250             switch (turtle_pack[i].state)
01251             {
01252             case TURTLE_STATE_SURFACE:
01253                 if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_SURFACE))
01254                     turtle_pack[i].fx.frame++;
01255 
01256                 // si no esta inicializado, inicializo timeout
01257                 if (!turtle_pack[i].fx.timeout)
01258                     turtle_pack[i].fx.timeout = get_rand_between(TURTLES_SURFACE_FRAMES_MIN,
    TURTLES_SURFACE_FRAMES_MAX);
01259 
01260                 // pasa a agua
01261                 if (!(turtle_pack[i].fx.cont % turtle_pack[i].fx.timeout))
01262                 {
01263                     turtle_pack[i].state = TURTLE_STATE_GOING_DOWN;
01264                     turtle_pack[i].fx.frame = 7;
01265                     turtle_pack[i].fx.timeout = 0;
01266                     turtle_pack[i].fx.cont = 1;
01267                 }
01268 
01269                 // Reinicia animacion
01270                 else if (turtle_pack[i].fx.frame == 7)
01271                     turtle_pack[i].fx.frame = 0;
01272 
01273                 break;
01274 
01275             case TURTLE_STATE_GOING_DOWN:
01276                 if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_GOING_DOWN))
01277                     turtle_pack[i].fx.frame++;
01278 
01279                 if (turtle_pack[i].fx.frame == 9)
01280                 {
01281                     turtle_pack[i].state = TURTLE_STATE_WATER;
01282                     turtle_pack[i].fx.cont = 1;
```

```
01283                 }
01284
01285                 break;
01286
01287             case TURTLE_STATE_WATER:
01288                 if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_WATER))
01289                     turtle_pack[i].fx.frame++;
01290
01291                 // si no esta inicializado, inicializo timeout
01292                 if (!turtle_pack[i].fx.timeout)
01293                     turtle_pack[i].fx.timeout = get_rand_between(TURTLES_WATER_FRAMES_MIN,
      TURTLES_WATER_FRAMES_MAX);
01294
01295                 // pasa a fuera
01296                 if (!(turtle_pack[i].fx.cont % turtle_pack[i].fx.timeout))
01297                 {
01298                     turtle_pack[i].state = TURTLE_STATE_GOING_UP;
01299                     turtle_pack[i].fx.frame = 10;
01300                     turtle_pack[i].fx.timeout = 0;
01301                     turtle_pack[i].fx.cont = 1;
01302                 }
01303
01304                 // Reinicia animacion
01305                 else if (turtle_pack[i].fx.frame == 11)
01306                     turtle_pack[i].fx.frame = 9;
01307
01308                 break;
01309
01310             case TURTLE_STATE_GOING_UP:
01311                 if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_GOING_UP))
01312                     turtle_pack[i].fx.frame--;
01313
01314                 if (turtle_pack[i].fx.frame == 7)
01315                 {
01316                     turtle_pack[i].fx.frame = 6;
01317                     turtle_pack[i].state = TURTLE_STATE_SURFACE;
01318                     turtle_pack[i].fx.cont = 1;
01319                 }
01320
01321                 break;
01322
01323             default:
01324                 break;
01325             }
01326
01327             // chequea si llego a los limites
01328             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))
01329             {
01330                 turtle_pack[i].used = false;
01331                 used--;
01332             }
01333
01334             // printf("~turtle_pack%d lane%d dx%d~\n", i, turtle_pack[i].lane, turtle_pack[i].dx);
01335         }
01336     }
01337 }
01338
01339 static void turtles_draw(void)
01340 {
01341     int i, j, flag;
01342     for (i = 0; i < TURTLES_MAX_USED; i++)
01343     {
01344         if (turtle_pack[i].used)
01345         {
01346             for (j = 0; j < turtle_pack[i].turtles_in_pack; j++)
01347             {
01348                 if (turtle_pack[i].dx < 0)
01349                     flag = ALLEGRO_FLIP_HORIZONTAL;
01350                 else
01351                     flag = 0;
01352
01353                 al_draw_bitmap(sprites.turtle[turtle_pack[i].fx.frame], turtle_pack[i].x + TURTLE_SIDE
      * j, turtle_pack[i].y, flag);
01354             }
01355
01356 #ifdef DEBUG_ENTITIES_TEXT
01357             // Dibujo hitbox
01358             allegro_draw_hitbox(turtle_pack[i].x, turtle_pack[i].y, turtle_pack[i].wide, TURTLE_SIDE);
01359 #endif
01360         }
01361     }
01362
01363 #ifdef DEBUG_ENTITIES_TEXT
01364     // coordenadas
01365     int space;
01366     for (i = 0, space = 350; i < TURTLES_MAX_USED; i++, space += 10)
```

```
01367     {
01368         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);
01369     }
01370 #endif
01371 }
01372
01373 static void coin_init(void)
01374 {
01375     coin.used = false;
01376     coin.y = CELL_H + SPRITE_COIN_OFFSET_XY + GOAL_ROW_OFFSET_Y_FIX;
01377
01378     coin.fx.blink_timer = 0;
01379     coin.fx.timeout = 0;
01380     coin.fx.flag = false;
01381     coin.fx.cont = 1;
01382 }
01383
01384 static void coin_update(void)
01385 {
01386     if (!coin.used)
01387     {
01388         // si no esta inicializado, inicializo timeout para spawneo
01389         if (!coin.fx.timeout)
01390             coin.fx.timeout = get_rand_between(COIN_SPAWN_FRAMES_MIN, COIN_SPAWN_FRAMES_MAX);
01391
01392         if (!(coin.fx.cont % coin.fx.timeout))
01393         {
01394             // calculo de coordenada x para alguno de los puntos de llegada
01395             int temp_goal = get_rand_between(0, MAX_GOALS - 1);
01396
01397             // si el goal está libre...
01398             if (!game_data_get_goal_state(temp_goal))
01399             {
01400                 allegro_sound_play_effect_coin_drop();
01401
01402                 coin.x = CELL_W * goal_cols[temp_goal] + SPRITE_COIN_OFFSET_XY - 1;
01403                 // marcado como usado
01404                 coin.used = true;
01405                 // desinicializo el timeout
01406                 coin.fx.timeout = 0;
01407
01408                 coin.fx.blink_timer = 0;
01409                 coin.fx.cont = 1;
01410                 coin.fx.frame_cont = 0;
01411             }
01412
01413             // si no, cuando pasa otro timeout se intenta de nuevo
01414             else
01415             {
01416             }
01417         }
01418     }
01419
01420     else
01421     {
01422         // timeout para despawneo
01423         if (!coin.fx.timeout)
01424             coin.fx.timeout = get_rand_between(COIN_DESPAWN_FRAMES_MIN, COIN_DESPAWN_FRAMES_MAX);
01425
01426         if (++coin.fx.blink_timer > coin.fx.timeout - COIN_FRAMES_TO_WARN_A)
01427         {
01428             if (coin.fx.blink_timer > coin.fx.timeout - COIN_FRAMES_TO_WARN_B)
01429             {
01430                 if (!(coin.fx.cont % COIN_WARNING_FRAMES_B))
01431                     coin.fx.flag = !coin.fx.flag;
01432             }
01433             else
01434             {
01435                 if (!(coin.fx.cont % COIN_WARNING_FRAMES_A))
01436                     coin.fx.flag = !coin.fx.flag;
01437             }
01438         }
01439
01440         if (!(game_frames % COIN_FRAME_RATE))
01441         {
01442             if (++coin.fx.frame_cont == SPRITE_COIN_FRAMES)
01443                 coin.fx.frame_cont = 0;
01444         }
01445
01446         // si se puede despawnear
01447         if (!(coin.fx.cont % coin.fx.timeout))
01448         {
01449             // coin no usada
01450             coin.used = false;
01451
01452             // desinicializo timeout
```

```
01453             coin.fx.timeout = 0;
01454
01455             // saco el blinking
01456             coin.fx.flag = false;
01457
01458             coin.fx.cont = 1;
01459         }
01460     }
01461
01462     coin.fx.cont++;
01463 }
01464
01465 static void coin_draw(void)
01466 {
01467     if (coin.used)
01468     {
01469         // Si no está el flag, dibujo sprite normalmente
01470         if (!coin.fx.flag)
01471             al_draw_bitmap(sprites.coin.frame[coin.fx.frame_cont], coin.x, coin.y, 0);
01472     }
01473
01474 #ifdef DEBUG_ENTITIES_TEXT
01475
01476     // hitbox
01477     allegro_draw_hitbox(coin.x, coin.y, COIN_SIDE, COIN_SIDE);
01478
01479     // coordenadas
01480     int space = 500;
01481     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);
01482 #endif
01483 }
01484
01485 static void fix_frog_coord_y(void)
01486 {
01487     int y = (frog.y - FROG_OFFSET_Y);
01488
01489     int y_values[ROWS];
01490
01491     int i;
01492
01493     // Carga valores "correctos" de y
01494     for (i = 1; i < ROWS - 1; i++)
01495         y_values[i] = i * CELL_H;
01496
01497     int temp_a, temp_b;
01498     for (i = 1; i < ROWS - 1; i++)
01499     {
01500         temp_a = y - y_values[i];
01501
01502         if (temp_a > 0)
01503             continue;
01504         if (temp_a == 0)
01505             break;
01506
01507         temp_b = y_values[i - 1] - y;
01508
01509         //"si está más cerca de la fila 'i' que de la 'i+1"
01510         if (temp_a <= temp_b)
01511             frog.y = y_values[i - 1] + FROG_OFFSET_Y;
01512         else
01513             frog.y = y_values[i] + FROG_OFFSET_Y;
01514
01515         break;
01516     }
01517 }
01518
01519 static bool is_frog_in_goal(void)
01520 {
01521     bool state = false;
01522     int x = frog.x;
01523
01524     int i, x_col;
01525     for (i = 0; i < MAX_GOALS; i++)
01526     {
01527         // Coordenada top left del punto de llegada
01528         x_col = goal_cols[i] * CELL_W;
01529
01530         // Calculo para ver si entro bien o no
01531         if ((x > x_col - GOAL_ROW_MARGIN_TO_REACH) &&
01532             ((x + FROG_W) < x_col + CELL_W + GOAL_ROW_MARGIN_TO_REACH))
01533         {
01534             // coodenada X aceptable
01535             state = true;
01536             break;
01537         }
01538     }
```

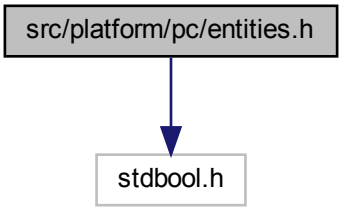
```
01539
01540     // Si coincide en coordenada y el goal esta libre...
01541     if (state && !game_data_get_goal_state(i))
01542     {
01543         // marca el goal como completo
01544         game_data_set_goal(i);
01545     }
01546     else
01547     {
01548         // no llego a un goal valido
01549         state = false;
01550     }
01551
01552     return state;
01553 }
01554
01555 static void corpse_init(int x, int y)
01556 {
01557     corpse_fx.flag = true;
01558     corpse_fx.timer = 1;
01559     corpse_fx.x = x - FROG_OFFSET_X + SPRITE_DEAD_OFFSET;
01560     corpse_fx.y = y - FROG_OFFSET_Y + SPRITE_DEAD_OFFSET;
01561 }
01562
01563 static void corpse_update(void)
01564 {
01565     if (corpse_fx.flag)
01566     {
01567         if (!(corpse_fx.timer++ % SPRITE_DEAD_TIMEOUT))
01568             corpse_fx.flag = false;
01569     }
01570 }
01571
01572 static void corpse_draw(void)
01573 {
01574     if (corpse_fx.flag)
01575         al_draw_bitmap(sprites.dead, corpse_fx.x, corpse_fx.y, 0);
01576 }
01577
01578 static void splash_init(int x, int y)
01579 {
01580     splash_fx.flag = true;
01581     splash_fx.cont = 1;
01582     splash_fx.frame_cont = 0;
01583     splash_fx.x = x - FROG_OFFSET_X + SPRITE_SPLASH_OFFSET_X;
01584     splash_fx.y = y - FROG_OFFSET_Y + SPRITE_SPLASH_OFFSET_Y;
01585 }
01586
01587 static void splash_update(void)
01588 {
01589     if (splash_fx.flag)
01590     {
01591         if (!(splash_fx.cont % SPRITE_SPLASH_RATE))
01592         {
01593             if (++splash_fx.frame_cont == SPRITE_SPLASH_FRAMES)
01594             {
01595                 splash_fx.frame_cont = 0;
01596                 splash_fx.flag = false;
01597             }
01598         }
01599
01600         splash_fx.cont++;
01601     }
01602 }
01603
01604 static void splash_draw(void)
01605 {
01606     if (splash_fx.flag)
01607         al_draw_bitmap(sprites.splash.frame[splash_fx.frame_cont], splash_fx.x, splash_fx.y, 0);
01608 }
```

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

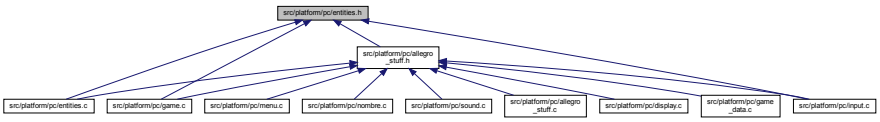
Header del modulo entities. Prototipos de funciones globales para el manejo de entidades.

```
#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.
- void entities_set_tutorial (void)

  Setea el estado del flag tutorial.
- bool entities_get_tutorial (void)

  Indica el estado del flag tutorial.

4.25.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

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 416 of file entities.c.

4.25.2.2 entities_get_tutorial() `bool entities_get_tutorial ( void )`

Indica el estado del flag tutorial.

Returns

true Tutorial done

false Playing tutorial

Definition at line 445 of file entities.c.

4.25.2.3 entities_init() `void entities_init ( void )`

Inicializa las entidades.

Definition at line 386 of file entities.c.

4.25.2.4 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 430 of file entities.c.

4.25.2.5 entities_set_tutorial() void entities_set_tutorial (void)

Setea el estado del flag tutorial.

Definition at line 441 of file entities.c.

4.25.2.6 entities_update() void entities_update (void)

Actualiza las entidades.

Definition at line 402 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
00054 void entities_set_tutorial(void);
00055
00062 bool entities_get_tutorial(void);
00063
00064 /*****************************************************************************
00065  *****************************************************************************/
00066
00067 #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 197 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
00158
00159     if (allegro_get_last_key() == ALLEGRO_KEY_8 && !allegro_get_rick_flag())
00160     {
00161         allegro_rick_on();
00162         allegro_set_rick_flag(true);
00163     }
00164
00165     if (allegro_get_last_key() == ALLEGRO_KEY_9 && allegro_get_rick_flag())
00166     {
00167         allegro_rick_off();
00168         allegro_set_rick_flag(false);
00169     }
00170
00171     if (allegro_get_var_redraw())
00172     {
00173         allegro_clear_display();
00174         allegro_draw_background();
00175
00176         if (allegro_get_rick_flag())
00177             allegro_rick_draw();
00178
00179         entities_draw();
00180         game_data_draw();
```

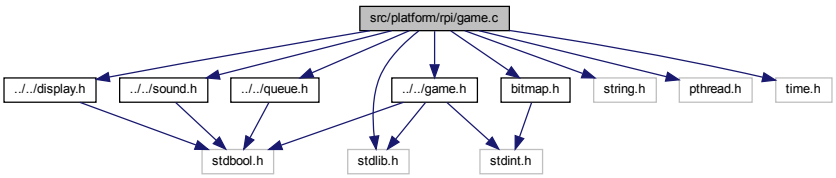
```
00181
00182         al_draw_bitmap(sprites.border, SPRITE_BORDER_START_X, SPRITE_BORDER_START_Y, 0);
00183
00184         if(!entities_get_tutorial()){
00185                 al_draw_bitmap(sprites.tutorial, 0, 0, 0);
00186             }
00187         if(!entities_get_tutorial() && allegro_get_last_key() == ALLEGRO_KEY_SPACE){
00188                 entities_set_tutorial();
00189             }
00190
00191         al_flip_display();
00192
00193         allegro_set_var_redraw(false);
00194     }
00195 }
00196
00197 void reanudarJuego(void)
00198 {
00199 }
```

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 SPAWN_MOVIMIENTOS 64
- #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

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 32 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.

4.29.2.5 SPAWN_MOVIMIENTOS `#define SPAWN_MOVIMIENTOS 64`

Definition at line 31 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 340 of file game.c.

4.29.3.2 actualizarMapa() `void actualizarMapa ( )`

Definition at line 145 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 200 of file game.c.

4.29.3.4 getNivel() `int getNivel ( void )`

Devuelve el nivel//run del jugador.

Returns

int

Definition at line 205 of file game.c.

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

Devuelve el nombre del jugador.

Returns

char*

Definition at line 191 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 195 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 313 of file game.c.

4.29.3.8 limpiarMapa() `void limpiarMapa ( )`

Definition at line 111 of file game.c.

4.29.3.9 moverAdelante() `void moverAdelante ( void )`

Avanza el jugador.

Definition at line 242 of file game.c.

4.29.3.10 moverAtras() `void moverAtras (`

`void )`

Retrocede el jugador.

Definition at line 276 of file game.c.

4.29.3.11 moverCarrriles() `void moverCarrriles (`

`int x )`

Definition at line 115 of file game.c.

4.29.3.12 moverDcha() `void moverDcha (`

`void )`

Mueve el jugador a la derecha.

Definition at line 292 of file game.c.

4.29.3.13 moverIzda() `void moverIzda (`

`void )`

Mueve el jugador a la izquierda.

Definition at line 282 of file game.c.

4.29.3.14 pausarJuego() `void pausarJuego (`

`void )`

Pausa el juego.

Definition at line 334 of file game.c.

4.29.3.15 perderVida() `void perderVida (`

`void )`

Resta una vida.

Definition at line 302 of file game.c.

4.29.3.16 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 375 of file game.c.

4.29.3.17 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 164 of file game.c.

4.29.3.18 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 210 of file game.c.

4.29.3.19 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 219 of file game.c.

4.29.3.20 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 186 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 106 of file game.c.

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

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 101 of file game.c.

4.29.3.23 spawnearAutos() `void spawnearAutos ( )`

Definition at line 123 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 181 of file game.c.

4.29.4 Variable Documentation

4.29.4.1 agua `bool agua`

Definition at line 52 of file game.c.

4.29.4.2 completo `uint32_t completo`

Definition at line 66 of file game.c.

4.29.4.3 derecho `uint16_t derecho`

Definition at line 69 of file game.c.

4.29.4.4 dificultad `int dificultad`

Definition at line 41 of file game.c.

4.29.4.5 disp_matriz `matriz_t disp_matriz [extern]`

Definition at line 27 of file display.c.

4.29.4.6 izquierdo `uint16_t izquierdo`

Definition at line 70 of file game.c.

4.29.4.7 jugador_1 `uint16_t jugador_1`

Definition at line 47 of file game.c.

4.29.4.8 jugador_2 `uint16_t jugador_2`

Definition at line 48 of file game.c.

4.29.4.9 jugador_posicion_oeste `int jugador_posicion_oeste`

Definition at line 44 of file game.c.

4.29.4.10 jugador_posicion_sur `int jugador_posicion_sur`

Definition at line 43 of file game.c.

4.29.4.11 jugando `bool jugando`

Definition at line 51 of file game.c.

4.29.4.12 mapa `matriz_t mapa`

Definition at line 63 of file game.c.

4.29.4.13 max_puntos `unsigned long long max_puntos`

Definition at line 50 of file game.c.

4.29.4.14 niv_actual `int niv_actual`

Definition at line 42 of file game.c.

4.29.4.15 nombre_jugador `char nombre_jugador[L_MAX]`

Definition at line 40 of file game.c.

4.29.4.16 pre_timeout `bool pre_timeout`

Definition at line 55 of file game.c.

4.29.4.17 puntos `unsigned long long puntos`

Definition at line 49 of file game.c.

4.29.4.18 ranas `uint16_t ranas`

Definition at line 46 of file game.c.

4.29.4.19 refresco_autos `bool refresco_autos`

Definition at line 54 of file game.c.

4.29.4.20 refresco_jugador `bool refresco_jugador`

Definition at line 53 of file game.c.

4.29.4.21 tiempo `clock_t tiempo`

Definition at line 57 of file game.c.

4.29.4.22 tiempo_alerta `clock_t tiempo_alerta`

Definition at line 62 of file game.c.

4.29.4.23 tiempo_inicio `clock_t tiempo_inicio`

Definition at line 58 of file game.c.

4.29.4.24 tiempo_referencia `clock_t tiempo_referencia`

Definition at line 59 of file game.c.

4.29.4.25 tiempo_refresco_autos `clock_t tiempo_refresco_autos`

Definition at line 61 of file game.c.

4.29.4.26 tiempo_refresco_jugador `clock_t tiempo_refresco_jugador`

Definition at line 60 of file game.c.

4.29.4.27 timeout `bool timeout`

Definition at line 56 of file game.c.

4.29.4.28 vidas `uint16_t vidas`

Definition at line 45 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 SPAWN_MOVIMIENTOS 64
00032 #define L_MAX 64
00033
00034 /*******************************************************************************
00035  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00036  ******************************************************************************/
00037
00038 static struct
00039 {
00040   char nombre_jugador[L_MAX];
00041   int dificultad;
00042   int niv_actual;
00043   int jugador_posicion_sur;
00044   int jugador_posicion_oeste;
00045   uint16_t vidas;
00046   uint16_t ranas;
00047   uint16_t jugador_1;
00048   uint16_t jugador_2;
00049   unsigned long long puntos;
00050   unsigned long long max_puntos;
00051   bool jugando;
00052   bool agua;
00053   bool refresco_jugador;
00054   bool refresco_autos;
00055   bool pre_timeout; // para avisar si queda poco tiempo
00056   bool timeout;
00057   clock_t tiempo;
00058   clock_t tiempo_inicio;
00059   clock_t tiempo_referencia;
```

```
00060   clock_t tiempo_refresco_jugador;
00061   clock_t tiempo_refresco_autos;
00062   clock_t tiempo_alerta;
00063   matriz_t mapa;
00064   union
00065   {
00066     uint32_t completo;
00067     struct
00068     {
00069       uint16_t derecho;
00070       uint16_t izquierdo;
00071     };
00072   } carril[CANT_CARRILES];
00073 } juego;
00074
00075 /*****************************************************************************
00076  * VARIABLES WITH GLOBAL SCOPE
00077  *****************************************************************************/
00078
00079 extern matriz_t disp_matriz;
00080
00081 /*****************************************************************************
00082  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00083  *****************************************************************************/
00084
00085 static void reiniciarTimer();
00086
00087 /*****************************************************************************
00088  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00089  *****************************************************************************/
00090
00091 static pthread_t ttiempo;
00092
00093 static void *threadTiempo(void *ptr);
00094
00095 /*****************************************************************************
00096  *****************************************************************************
00097                         GLOBAL FUNCTION DEFINITIONS
00098  *****************************************************************************
00099  *****************************************************************************/
00100
00101 void setNombre(char *nombre)
00102 {
00103   strcpy(juego.nombre_jugador, nombre);
00104 }
00105
00106 void setMaxPuntos(unsigned long long max)
00107 {
00108   juego.max_puntos = max;
00109 }
00110
00111 void limpiarMapa()
00112 {
00113 }
00114
00115 void moverCarrriles(int x)
00116 {
00117   for (int i = 0; i < 5; i++)
00118     juego.carril[i].completo «= x; // movimiento de autos/troncos
00119   if(juego.agua && juego.jugador_posicion_sur < CANT_FILAS - 2 && juego.jugador_posicion_sur > 4)
00120     moverIzda(); //animación de movimiento de jugador con los troncos
00121 }
00122
00123 void spawnearAutos()
00124 {
00125   int i;
00126   for (i = 0; i < 5; i++)
00127   {
00128     if (juego.agua)
00129     {
00130       if (!(juego.carril[i].completo & 0b1111111111) && !(rand() % 15))
00131         juego.carril[i].completo |= 0b111111;
00132       else if (!(juego.carril[i].completo & 0b11111111111) && !(rand() % 30))
00133         juego.carril[i].completo |= 0b11111111;
00134     }
00135     else
00136     {
00137       if (!(juego.carril[i].completo & 0b111111) && !(rand() % 15))
00138         juego.carril[i].completo |= 0b11;
00139       else if (!(juego.carril[i].completo & 0b11111111) && !(rand() % 30))
00140         juego.carril[i].completo |= 0b1111;
00141     }
00142   }
00143 }
00144
00145 void actualizarMapa()
00146 {
```

```
00147   if (juego.agua)
00148   {
00149     juego.mapa[2] = juego.ranas;
00150     juego.mapa[3] = juego.ranas;
00151   }
00152   else
00153   {
00154     juego.mapa[2] = 0;
00155     juego.mapa[3] = 0;
00156   }
00157   for (int i = 0; i < 5; i++)
00158   {
00159     juego.mapa[POS_AUTOS_INICIO + 2 * i] = juego.carril[i].izquierdo;
00160     juego.mapa[POS_AUTOS_INICIO + 2 * i + 1] = juego.carril[i].izquierdo;
00161   }
00162 }
00163
00164 void refrescar()
00165 {
00166   if (juego.refresco_autos)
00167   {
00168     moverCarrriles(1);
00169     spawnearAutos();
00170     juego.refresco_autos = false;
00171   }
00172   if (juego.refresco_jugador)
00173   {
00174     uint16_t tmp = juego.jugador_1;
00175     juego.jugador_1 = juego.jugador_2;
00176     juego.jugador_2 = tmp;
00177     juego.refresco_jugador = false;
00178   }
00179 }
00180
00181 bool tiempoRefrescoEntidades(void)
00182 {
00183   return juego.refresco_jugador || juego.refresco_autos;
00184 }
00185
00186 void setDificultad(int dificultad)
00187 {
00188   juego.dificultad = dificultad;
00189 }
00190
00191 char *getNombre()
00192 {
00193   return juego.nombre_jugador;
00194 }
00195 unsigned long long getPuntos()
00196 {
00197   return juego.puntos;
00198 }
00199
00200 unsigned long long getMaxPuntos()
00201 {
00202   return juego.max_puntos;
00203 }
00204
00205 int getNivel()
00206 {
00207   return juego.niv_actual;
00208 }
00209
00210 void reiniciarNivel()
00211 {
00212   juego.ranas = 0b1001001001001001;
00213   juego.agua = false;
00214   respawn();
00215   reiniciarTimer();
00216   reanudarJuego();
00217 }
00218
00219 void respawn()
00220 {
00221   juego.jugador_posicion_sur = CANT_FILAS - 1;
00222   if (!juego.agua)
00223   {
00224     juego.jugador_1 = 0b0000000100000000;
00225     juego.jugador_2 = 0b0000000010000000;
00226     juego.jugador_posicion_oeste = 7;
00227   }
00228
00229   for (int i = 0; i < CANT_CARRILES; i++)
00230     juego.carril[i].completo = 0;
00231   limpiarMatriz(juego.mapa);
00232
00233   for (int i = 0; i < SPAWN_MOVIMIENTOS; i++)
```

```
00234   {
00235     moverCarrriles(1);
00236     spawnearAutos();
00237   }
00238
00239   actualizarMapa();
00240 }
00241
00242 void moverAdelante()
00243 {
00244   if (juego.jugador_posicion_sur > 3)
00245     juego.jugador_posicion_sur--;
00246   else
00247   {
00248     if (!juego.agua) //si voy por la mitad de la ronda
00249     {
00250       juego.agua = true;
00251       respawn();
00252     }
00253     else
00254     {
00255       juego.puntos += 500 + 250*(juego.dificultad);
00256       juego.ranas |= juego.jugador_1 | juego.jugador_2;
00257       if (juego.ranas == 0b1111111111111111) //si completé el nivel
00258       {
00259         juego.puntos += 1000 + 250*(juego.dificultad);
00260         pausarJuego();
00261         juego.niv_actual++;
00262         reproducirEfecto(EFECTO_NIVEL_COMPLETO);
00263         reiniciarNivel();
00264       }
00265       else //si completé la ronda pero aún falta completar el nivels
00266       {
00267         juego.agua = false;
00268         reproducirEfecto(EFECTO_META);
00269         respawn();
00270         juego.tiempo += CLOCKS_PER_SEC * 10;
00271       }
00272     }
00273   }
00274 }
00275
00276 void moverAtras()
00277 {
00278   if (juego.jugador_posicion_sur < 15)
00279     juego.jugador_posicion_sur++;
00280 }
00281
00282 void moverIzda()
00283 {
00284   if (juego.jugador_posicion_oeste > 0)
00285   {
00286     juego.jugador_posicion_oeste--;
00287     juego.jugador_1 «= 1;
00288     juego.jugador_2 «= 1;
00289   }
00290 }
00291
00292 void moverDcha()
00293 {
00294   if (juego.jugador_posicion_oeste < 14)
00295   {
00296     juego.jugador_posicion_oeste++;
00297     juego.jugador_1 »= 1;
00298     juego.jugador_2 »= 1;
00299   }
00300 }
00301
00302 void perderVida()
00303 {
00304   juego.agua ? reproducirEfecto(EFECTO_AHOGADO) : reproducirEfecto(EFECTO_IMPACTO);
00305   juego.agua = false;
00306   juego.vidas «= 1;
00307   if (!juego.vidas)
00308     queueInsertar(GAME_OVER);
00309   else
00310     respawn();
00311 }
00312
00313 void inicializarJuego()
00314 {
00315   juego.puntos = 0;
00316   juego.niv_actual = 1;
00317   juego.vidas = 0b1110000000000000;
00318 }
00319
00320 void reiniciarTimer()
```

```
00321 {
00322   juego.tiempo_inicio = CLOCKS_PER_SEC * (150 - (2 * (juego.dificultad + juego.niv_actual)));
00323   juego.tiempo = juego.tiempo_inicio;
00324   juego.tiempo_referencia = juego.tiempo_inicio;
00325   juego.tiempo_refresco_autos = CLOCKS_PER_SEC * (1 - 0.05 * (2 * (juego.dificultad + juego.niv_actual
      - 1)));
00326   juego.tiempo_refresco_jugador = CLOCKS_PER_SEC » 1;
00327   juego.tiempo_alerta = CLOCKS_PER_SEC * 15;
00328   juego.refresco_autos = false;
00329   juego.refresco_jugador = false;
00330   juego.timeout = false;
00331   juego.pre_timeout = false;
00332 }
00333
00334 void pausarJuego()
00335 {
00336   juego.jugando = false;
00337   pthread_join(ttiempo, NULL);
00338 }
00339
00340 void actualizarInterfaz()
00341 {
00342   actualizarMapa();
00343   copiarMatriz(disp_matriz, juego.mapa);
00344   disp_matriz[0] = juego.vidas;
00345
00346   // algoritmo para convertir el tiempo al indicador de 16 bits por aproximaciones sucesivas
00347   clock_t acc = 0, frac = juego.tiempo_inicio » 1;
00348   uint16_t tiempo_bits = 0;
00349   int bit_de_referencia = 0b0000000000000001;
00350   int j;
00351   for (j = 8; j; j »= 1, frac »= 1)
00352   {
00353     if (juego.tiempo > (acc + frac))
00354     {
00355       acc += frac;
00356       bit_de_referencia «= j;
00357       tiempo_bits |= bit_de_referencia - 1; // hago 1 todos los bits menos significativos que el de
      referencia
00358     }
00359   }
00360
00361   disp_matriz[1] = tiempo_bits;
00362
00363   disp_matriz[(juego.jugador_posicion_sur) - 1] |= juego.jugador_1;
00364   disp_matriz[juego.jugador_posicion_sur] |= juego.jugador_2;
00365
00366   actualizarDisplay();
00367
00368   if ((juego.mapa[(juego.jugador_posicion_sur) - 1]) & juego.jugador_1 ||
      (juego.mapa[(juego.jugador_posicion_sur)]) & juego.jugador_2)
00369     perderVida();
00370
00371   if (juego.timeout)
00372     queueInsertar(GAME_OVER);
00373 }
00374
00375 void reanudarJuego(void)
00376 {
00377   juego.jugando = true;
00378   pthread_create(&ttiempo, NULL, threadTiempo, NULL);
00379 }
00380
00381 /*****************************************************************************
00382  *****************************************************************************
00383                          LOCAL FUNCTION DEFINITIONS
00384  *****************************************************************************
00385  *****************************************************************************/
00386
00387 static void *threadTiempo(void *ptr)
00388 {
00389   clock_t ref = clock();
00390   clock_t ref_autos = ref + juego.tiempo_refresco_autos, ref_jugador = ref +
      juego.tiempo_refresco_jugador;
00391   while (juego.jugando)
00392   {
00393     if (!(juego.timeout))
00394     {
00395       clock_t tiempo = clock();
00396       juego.tiempo = juego.tiempo_referencia - (tiempo - ref);
00397       juego.timeout = juego.tiempo <= 0;
00398       if (!juego.pre_timeout && juego.tiempo <= juego.tiempo_alerta)
00399       {
00400         juego.pre_timeout = true;
00401         reproducirEfecto(EFECTO_POCO_TIEMPO);
00402       }
00403       if (tiempo > ref_autos)
```

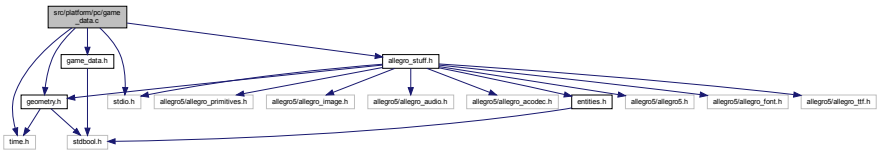
```
00404       {
00405         juego.refresco_autos = true;
00406         ref_autos = tiempo + juego.tiempo_refresco_autos;
00407       }
00408       if (tiempo > ref_jugador)
00409       {
00410         juego.refresco_jugador = true;
00411         ref_jugador = tiempo + juego.tiempo_refresco_jugador;
00412       }
00413     }
00414     else
00415       ref = clock();
00416   }
00417
00418   juego.tiempo_referencia = juego.tiempo;
00419
00420   return NULL;
00421 }
```

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 60
- #define RUN_TIME_LEFT_REDUCE_FACTOR_A 2
- #define RUN_TIME_LEFT_REDUCE_LIMIT_A 5
- #define RUN_TIME_LEFT_REDUCE_FACTOR_B 5
- #define RUN_TIME_LEFT_REDUCE_UNTIL 10
- #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 41 of file game_data.c.

4.31.2.2 EXTRA_TIME_PER_GOAL `#define EXTRA_TIME_PER_GOAL 10`

Definition at line 40 of file game_data.c.

4.31.2.3 HUD_EXTRA_INFO_RATE `#define HUD_EXTRA_INFO_RATE 1`

Definition at line 46 of file game_data.c.

4.31.2.4 HUD_EXTRA_INFO_TIMING `#define HUD_EXTRA_INFO_TIMING 120`

Definition at line 45 of file game_data.c.

4.31.2.5 INITIAL_RUN_TIME_LEFT `#define INITIAL_RUN_TIME_LEFT 60`

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 RUN_TIME_LEFT_REDUCE_FACTOR_A `#define RUN_TIME_LEFT_REDUCE_FACTOR_A 2`

Definition at line 35 of file game_data.c.

4.31.2.9 RUN_TIME_LEFT_REDUCE_FACTOR_B `#define RUN_TIME_LEFT_REDUCE_FACTOR_B 5`

Definition at line 37 of file game_data.c.

4.31.2.10 RUN_TIME_LEFT_REDUCE_LIMIT_A `#define RUN_TIME_LEFT_REDUCE_LIMIT_A 5`

Definition at line 36 of file game_data.c.

4.31.2.11 RUN_TIME_LEFT_REDUCE_UNTIL `#define RUN_TIME_LEFT_REDUCE_UNTIL 10`

Definition at line 38 of file game_data.c.

4.31.2.12 SCORE_PER_GOAL `#define SCORE_PER_GOAL 500`

Definition at line 30 of file game_data.c.

4.31.2.13 SCORE_PER_GOAL_COIN `#define SCORE_PER_GOAL_COIN 750`

Definition at line 31 of file game_data.c.

4.31.2.14 SCORE_PER_RUN `#define SCORE_PER_RUN 1000`

Definition at line 32 of file game_data.c.

4.31.2.15 TIME_LEFT_WARNING `#define TIME_LEFT_WARNING 10`

Definition at line 43 of file game_data.c.

4.31.3 Enumeration Type Documentation

4.31.3.1 DATA_FLAGS `enum DATA_FLAGS`

Definition at line 79 of file game_data.c.

4.31.3.2 HUD_EXTRAS `enum HUD_EXTRAS`

Definition at line 88 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 400 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 350 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 358 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 307 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 316 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 447 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 386 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 286 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 381 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 366 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 439 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 422 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 292 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 417 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 465 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 335 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 345 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 302 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 330 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 371 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 227 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 340 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 391 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 432 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 376 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 427 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 325 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 297 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 249 of file game_data.c.

4.31.5 Variable Documentation

4.31.5.1 flag bool flag

Definition at line 99 of file game_data.c.

4.31.5.2 shifter int shifter

Definition at line 102 of file game_data.c.

4.31.5.3 timer int timer

Definition at line 101 of file game_data.c.

4.31.5.4 value int value

Definition at line 100 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 60 //tiempo inicial de una partida
00035 #define RUN_TIME_LEFT_REDUCE_FACTOR_A 2 //reduccion A
00036 #define RUN_TIME_LEFT_REDUCE_LIMIT_A 5 //por cuantas runs usar la reduccion A
00037 #define RUN_TIME_LEFT_REDUCE_FACTOR_B 5 //reduccion B
00038 #define RUN_TIME_LEFT_REDUCE_UNTIL 10 //tiempo mínimo a tener
00039
00040 #define EXTRA_TIME_PER_GOAL 10       // 10s extras por llegar a una meta
00041 #define EXTRA_TIME_PER_BONUS_GOAL 15 // 15s extras por llegar a una meta con coin
00042
00043 #define TIME_LEFT_WARNING 10 // warning 10s antes del timeout
00044
00045 #define HUD_EXTRA_INFO_TIMING 120 // frames que duran los mensajes emergentes
00046 #define HUD_EXTRA_INFO_RATE 1     // "velocidad" de desplazamiento de mensajes
00047
00048 /*****************************************************************************
00049  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00050  *****************************************************************************/
00051
00052 typedef struct
00053 {
00054     int lives;
00055     unsigned long long score;
00056     unsigned long long score_max;
00057
00058     struct
00059     {
00060         int number;    // numero de run actual
00061         int time_left; // tiempo restante en la run
00062         int time;      // tiempo de la run actual
00063         long time_ref; // referencia de tiempo global de la run
00064     } run;
00065
00066     unsigned long frames;
00067     int timer_in_sec;
00068
00069     int difficulty;
00070
00071     char name[MAX_NAME_CHAR];
00072
00073     unsigned char flag;
00074
00075     bool goals[MAX_GOALS];
00076
00077 } data_t;
00078
00079 enum DATA_FLAGS
00080 {
00081     DATA_FLAG_STARTING,
00082     DATA_FLAG_NEXT_RUN,
00083     DATA_FLAG_TIME_EXCEEDED,
00084     DATA_FLAG_GAME_OVER
00085 };
00086
00087 // Tipos de mensajes emergentes en el HUD
00088 enum HUD_EXTRAS
00089 {
00090     HUD_EXTRA_TIME,
00091     HUD_EXTRA_SCORE,
00092     HUD_EXTRA_RUN,
00093     HUD_EXTRAS_MAX
00094 };
00095
00096 // Estructura para el manejo de mensajes emergentes en el HUD
00097 static struct
00098 {
00099     bool flag;   // Activado ó no
00100     int value;   // Valor extra a mostrar
00101     int timer;   // Contador interno para dejar de mostrar
00102     int shifter; // Contador interno para desplazar el mensaje verticalmente
00103 } hud_extra_stuff[HUD_EXTRAS_MAX];
00104
00105 /*****************************************************************************
00106  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00107  *****************************************************************************/
00108
00113 static void data_init(void);
00114
00119 static void data_update(void);
00120
00125 static void hud_draw(void);
00126
00131 static void draw_reached_goals(void);
00132
00137 static void next_run(void);
00138
```

```
00144 static void trigger_show_adding_time(int extra);
00145
00151 static void trigger_show_adding_score(int extra);
00152
00157 static void trigger_show_adding_run(void);
00158
00163 static void draw_extra_time(void);
00164
00169 static void draw_extra_score(void);
00170
00175 static void draw_extra_run(void);
00176
00181 static void progress_run_time_left(void);
00182
00189 static bool is_last_goal(void);
00190
00191 /*****************************************************************************
00192  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00193  *****************************************************************************/
00194
00195 // Datos internos del juego
00196 static data_t data;
00197
00198 // Referencia temporal del inicio del juego
00199 static long time_ref;
00200
00201 static int char_h; // altura de un caracter
00202 static int char_w; // ancho de un caracter
00203
00204 static ALLEGRO_COLOR text_color;
00205
00206 // Flag para triggear sonido de poco tiempo
00207 static bool flag_low_time_warning;
00208
00209 // Auxiliar para hacer acciones en los cambios de segundos
00210 static int last_loop_time;
00211
00212 // Tiempo restante inicial de una nueva run. Se puede modificar externamente
00213 static int new_run_time_left;
00214
00215 // Auxiliar para mostrar el score gradualmente en el HUD
00216 static unsigned long long score_display;
00217
00218 // Score maximo no actualizado en game over
00219 static unsigned long long max_score_no_updated;
00220
00221 /*****************************************************************************
00222  *****************************************************************************
00223             GLOBAL FUNCTION DEFINITIONS
00224  *****************************************************************************
00225  *****************************************************************************/
00226
00227 void game_data_init(void)
00228 {
00229     time_ref = time(NULL);
00230
00231     char_h = allegro_get_var_font_h();
00232     char_w = allegro_get_var_font_w();
00233
00234     text_color = al_map_rgb(255, 255, 255);
00235
00236     flag_low_time_warning = false;
00237
00238     new_run_time_left = INITIAL_RUN_TIME_LEFT;
00239
00240     score_display = 0;
00241
00242     int i;
00243     for(i = 0; i < HUD_EXTRAS_MAX; i++)
00244         hud_extra_stuff[i].flag = false;
00245
00246     data_init();
00247 }
00248
00249 void game_data_update(void)
00250 {
00251     data_update();
00252
00253     if (data.flag == DATA_FLAG_NEXT_RUN)
00254     {
00255         next_run();
00256         data.flag = DATA_FLAG_STARTING;
00257     }
00258
00259     if (data.run.time_left == TIME_LEFT_WARNING && !flag_low_time_warning)
00260     {
00261         allegro_sound_play_effect_low_time();
```

```
00262
00263         flag_low_time_warning = true;
00264     }
00265     else if (data.run.time_left < TIME_LEFT_WARNING)
00266         flag_low_time_warning = false;
00267
00268     if (max_score_no_updated != data.score_max)
00269         max_score_no_updated = data.score_max;
00270
00271     if (data.run.time_left == 0)
00272     {
00273         // data.flag = DATA_FLAG_TIME_EXCEEDED;
00274         game_data_subtract_live();
00275         allegro_sound_play_effect_no_time();
00276         data.flag = DATA_FLAG_GAME_OVER;
00277     }
00278
00279     if (data.lives == 0)
00280     {
00281         data.flag = DATA_FLAG_GAME_OVER;
00282     }
00283
00284 }
00285
00286 void game_data_draw(void)
00287 {
00288     hud_draw();
00289     draw_reached_goals();
00290 }
00291
00292 int game_data_get_lives(void)
00293 {
00294     return (data.lives);
00295 }
00296
00297 void game_data_subtract_live(void)
00298 {
00299     data.lives--;
00300 }
00301
00302 unsigned long long game_data_get_score(void)
00303 {
00304     return (data.score);
00305 }
00306
00307 void game_data_add_score(void)
00308 {
00309     unsigned long long adder = SCORE_PER_GOAL + 250*(game_data_get_diff() - 1);
00310
00311     data.score += adder;
00312     if(!is_last_goal())
00313         trigger_show_adding_score(adder);
00314 }
00315
00316 void game_data_add_score_bonus(void)
00317 {
00318     unsigned long long adder = SCORE_PER_GOAL_COIN + 250*(game_data_get_diff() - 1);
00319
00320     data.score += adder;
00321     if(!is_last_goal())
00322         trigger_show_adding_score(adder);
00323 }
00324
00325 void game_data_set_score_max(unsigned long long score)
00326 {
00327     data.score_max = score;
00328 }
00329
00330 unsigned long long game_data_get_score_max(void)
00331 {
00332     return data.score_max;
00333 }
00334
00335 int game_data_get_run_number(void)
00336 {
00337     return (data.run.number);
00338 }
00339
00340 void game_data_next_run(void)
00341 {
00342     data.flag = DATA_FLAG_NEXT_RUN;
00343 }
00344
00345 int game_data_get_run_time_left(void)
00346 {
00347     return (data.run.time_left);
00348 }
```

```
00349
00350 void game_data_add_run_time_goal(void)
00351 {
00352     data.run.time_left += EXTRA_TIME_PER_GOAL;
00353
00354     if(!is_last_goal())
00355         trigger_show_adding_time(EXTRA_TIME_PER_GOAL);
00356 }
00357
00358 void game_data_add_run_time_goal_bonus(void)
00359 {
00360     data.run.time_left += EXTRA_TIME_PER_BONUS_GOAL;
00361
00362     if(!is_last_goal())
00363         trigger_show_adding_time(EXTRA_TIME_PER_BONUS_GOAL);
00364 }
00365
00366 unsigned long game_data_get_frames(void)
00367 {
00368     return (data.frames);
00369 }
00370
00371 int game_data_get_timer_in_sec(void)
00372 {
00373     return (data.timer_in_sec);
00374 }
00375
00376 void game_data_set_diff(int diff)
00377 {
00378     data.difficulty = diff;
00379 }
00380
00381 int game_data_get_diff(void)
00382 {
00383     return (data.difficulty);
00384 }
00385
00386 void game_data_clear_name(void)
00387 {
00388     memset(data.name, 0, MAX_NAME_CHAR);
00389 }
00390
00391 void game_data_overwrite_name(char *name)
00392 {
00393     int i;
00394     for(i = 0; name[i] != '\0'; i++)
00395         data.name[i] = name[i];
00396
00397     //strcpy(data.name, name);
00398 }
00399
00400 void game_data_add_name_letter(char letter)
00401 {
00402     int length = strlen(data.name);
00403
00404     if ((letter == ALLEGRO_KEY_BACKSPACE) && (length > 0))
00405     {
00406         data.name[length - 1] = 0;
00407     }
00408
00409     else if (letter >= ALLEGRO_KEY_A && letter <= ALLEGRO_KEY_Z && length < MAX_NAME_CHAR)
00410     {
00411         letter += '@';
00412         data.name[length] = letter;
00413         data.name[length + 1] = 0;
00414     }
00415 }
00416
00417 char *game_data_get_name(void)
00418 {
00419     return (data.name);
00420 }
00421
00422 bool game_data_get_goal_state(unsigned int goal)
00423 {
00424     return data.goals[goal];
00425 }
00426
00427 void game_data_set_goal(unsigned int goal)
00428 {
00429     data.goals[goal] = true;
00430 }
00431
00432 void game_data_reset_goals(void)
00433 {
00434     int i;
00435     for (i = 0; i < MAX_GOALS; i++)
```

```
00436         data.goals[i] = false;
00437 }
00438
00439 bool game_data_get_game_over_flag(void)
00440 {
00441     if (data.flag == DATA_FLAG_GAME_OVER)
00442         return true;
00443     else
00444         return false;
00445 }
00446
00447 bool game_data_are_goals_full(void)
00448 {
00449     bool state = true;
00450
00451     int i;
00452     for (i = 0; i < MAX_GOALS; i++)
00453     {
00454         // si alguno esa vacio...
00455         if (!data.goals[i])
00456         {
00457             state = false;
00458             break;
00459         }
00460     }
00461
00462     return state;
00463 }
00464
00465 unsigned long long game_data_get_old_max_score(void)
00466 {
00467     return max_score_no_updated;
00468 }
00469
00470 /*****************************************************************************
00471  *****************************************************************************
00472             LOCAL FUNCTION DEFINITIONS
00473  *****************************************************************************
00474  *****************************************************************************/
00475
00476 static void data_init(void)
00477 {
00478     data.frames = 0;
00479     data.lives = MAX_LIVES;
00480     data.run.number = 0;
00481     data.run.time_left = new_run_time_left;
00482     data.run.time = 0;
00483     data.run.time_ref = time(NULL);
00484     data.score = 0;
00485     data.timer_in_sec = 0;
00486
00487     data.flag = DATA_FLAG_STARTING;
00488
00489     last_loop_time = 0;
00490
00491     game_data_reset_goals();
00492 }
00493
00494 static void data_update(void)
00495 {
00496     data.frames++;
00497
00498     if(entities_get_tutorial())
00499     {
00500         // diferencia entre el tiempo actual y el de referencia
00501         data.timer_in_sec = time(NULL) - time_ref;
00502         data.run.time = time(NULL) - data.run.time_ref;
00503
00504         if (data.run.time > last_loop_time)
00505         {
00506             data.run.time_left--;
00507             last_loop_time++;
00508         }
00509         last_loop_time = data.run.time;
00510     }
00511
00512 }
00513
00514 static void hud_draw(void)
00515 {
00516     // Dibuja la puntuacion en pantalla.
00517
00518     int i;
00519
00520     // Graduacion del score a mostrar para que vaya incrementando de apoco
00521     if (score_display != data.score)
00522     {
```

```
00523         int shifter;
00524
00525         for (i = 2, shifter = 0; i > 0; i--)
00526         {
00527             shifter = 1 « i;
00528             if (score_display <= (data.score - shifter))
00529                 score_display += shifter;
00530         }
00531     }
00532
00533     al_draw_textf(
00534         allegro_get_var_font(),
00535         text_color,
00536         1, 1, // Arriba a la izquierda.
00537         0,
00538         "Score: %06lld", // 6 cifras (por ahi es mucho).
00539         score_display);
00540
00541     draw_extra_score();
00542
00543     // Dibuja el numero de vuelta.
00544     al_draw_textf(
00545         allegro_get_var_font(),
00546         text_color,
00547         1, CELL_H - char_h - 5, // Para que quede abaj de la puntuacion en pantalla.
00548         0,
00549         "Run: %02d", // 2 cifras. No me acuerdo si esta bien asi.
00550         data.run.number);
00551
00552     draw_extra_run();
00553
00554     // Segundos.
00555     al_draw_textf(
00556         allegro_get_var_font(),
00557         text_color,
00558         al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w, 1,
00559         0,
00560         "Played Time: %04d",
00561         data.timer_in_sec);
00562
00563     // Tiempo restante
00564     al_draw_textf(
00565         allegro_get_var_font(),
00566         text_color,
00567         al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w,
00568         CELL_H - char_h - 5,
00569         0,
00570         "Time Left: %03d",
00571         data.run.time_left);
00572
00573     draw_extra_time();
00574
00575     // Dibuja vidas.
00576     for (int i = 0; i < data.lives; i++) // No se si la rana tiene 'frog.lives' pero aca va el
      equivalente.
00577         al_draw_bitmap(
00578             sprites.heart,
00579             DISPLAY_W - 100 + SPRITE_SIZE_HEART * (data.lives - i - 1),
00580             (CELL_H - char_h - 5) / 2,
00581             0);
00582 }
00583
00584 static void draw_reached_goals(void)
00585 {
00586     int i;
00587     for (i = 0; i < MAX_GOALS; i++)
00588     {
00589         // si algun goal fue alcanzado...
00590         if (data.goals[i])
00591             al_draw_bitmap(sprites.frog[6],
00592                            goal_cols[i] * CELL_W + FROG_OFFSET_X - 1,
00593                            CELL_H + FROG_OFFSET_Y + GOAL_ROW_OFFSET_Y_FIX,
00594                            0);
00595     }
00596 }
00597
00598 static void next_run(void)
00599 {
00600     data.run.number++;
00601
00602     progress_run_time_left();
00603     data.run.time_left = new_run_time_left;
00604
00605     data.run.time = 0;
00606     data.run.time_ref = time(NULL);
00607
00608     unsigned long long adder = SCORE_PER_RUN + 250*(game_data_get_diff() - 1);
```

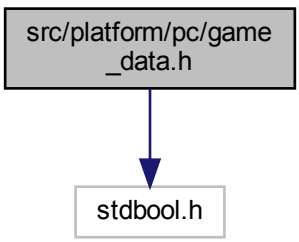
```
00609
00610     data.score += adder;
00611     trigger_show_adding_score(adder);
00612
00613     last_loop_time = 0;
00614
00615     flag_low_time_warning = false;
00616
00617     trigger_show_adding_run();
00618
00619     game_data_reset_goals();
00620 }
00621
00622 static void trigger_show_adding_time(int extra)
00623 {
00624     hud_extra_stuff[HUD_EXTRA_TIME].flag = true;
00625     hud_extra_stuff[HUD_EXTRA_TIME].value = extra;
00626     hud_extra_stuff[HUD_EXTRA_TIME].timer = 1;
00627     hud_extra_stuff[HUD_EXTRA_TIME].shifter = 0;
00628 }
00629
00630 static void trigger_show_adding_score(int extra)
00631 {
00632     hud_extra_stuff[HUD_EXTRA_SCORE].flag = true;
00633     hud_extra_stuff[HUD_EXTRA_SCORE].value = extra;
00634     hud_extra_stuff[HUD_EXTRA_SCORE].timer = 1;
00635     hud_extra_stuff[HUD_EXTRA_SCORE].shifter = 0;
00636 }
00637
00638 static void trigger_show_adding_run()
00639 {
00640     hud_extra_stuff[HUD_EXTRA_RUN].flag = true;
00641     hud_extra_stuff[HUD_EXTRA_RUN].value = 1;
00642     hud_extra_stuff[HUD_EXTRA_RUN].timer = 1;
00643     hud_extra_stuff[HUD_EXTRA_RUN].shifter = 0;
00644 }
00645
00646 static void draw_extra_time(void)
00647 {
00648     if (hud_extra_stuff[HUD_EXTRA_TIME].flag)
00649     {
00650         al_draw_textf(
00651             allegro_get_var_font(),
00652             text_color,
00653             al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w +
00654                 al_get_text_width(allegro_get_var_font(), "Time left: "),
00655             CELL_H - char_h - 5 + hud_extra_stuff[HUD_EXTRA_TIME].shifter,
00656             0,
00657             "%+3d",
00658             hud_extra_stuff[HUD_EXTRA_TIME].value);
00659
00660         hud_extra_stuff[HUD_EXTRA_TIME].shifter += HUD_EXTRA_INFO_RATE;
00661
00662         if (!(hud_extra_stuff[HUD_EXTRA_TIME].timer++ % HUD_EXTRA_INFO_TIMING))
00663             hud_extra_stuff[HUD_EXTRA_TIME].flag = false;
00664     }
00665 }
00666
00667 static void draw_extra_score(void)
00668 {
00669     if (hud_extra_stuff[HUD_EXTRA_SCORE].flag)
00670     {
00671         al_draw_textf(
00672             allegro_get_var_font(),
00673             text_color,
00674             1 + al_get_text_width(allegro_get_var_font(), "Score: "),
00675             1 + hud_extra_stuff[HUD_EXTRA_SCORE].shifter,
00676             0,
00677             "%+6d",
00678             hud_extra_stuff[HUD_EXTRA_SCORE].value);
00679
00680         hud_extra_stuff[HUD_EXTRA_SCORE].shifter += HUD_EXTRA_INFO_RATE;
00681
00682         if (!(hud_extra_stuff[HUD_EXTRA_SCORE].timer++ % HUD_EXTRA_INFO_TIMING))
00683             hud_extra_stuff[HUD_EXTRA_SCORE].flag = false;
00684     }
00685 }
00686
00687 static void draw_extra_run(void)
00688 {
00689     if (hud_extra_stuff[HUD_EXTRA_RUN].flag)
00690     {
00691         al_draw_textf(
00692             allegro_get_var_font(),
00693             text_color,
00694             1 + al_get_text_width(allegro_get_var_font(), "Run: "),
00695             CELL_H - char_h - 5 + hud_extra_stuff[HUD_EXTRA_RUN].shifter,
```

```
00696             0,
00697             "%+2d",
00698             hud_extra_stuff[HUD_EXTRA_RUN].value);
00699
00700         hud_extra_stuff[HUD_EXTRA_RUN].shifter += HUD_EXTRA_INFO_RATE;
00701
00702         if (!(hud_extra_stuff[HUD_EXTRA_RUN].timer++ % HUD_EXTRA_INFO_TIMING))
00703             hud_extra_stuff[HUD_EXTRA_RUN].flag = false;
00704     }
00705 }
00706
00707 static void progress_run_time_left(void)
00708 {
00709
00710     //Si la run es menor a la evaluada...
00711     if(data.run.number <= RUN_TIME_LEFT_REDUCE_LIMIT_A)
00712     {
00713         new_run_time_left -= RUN_TIME_LEFT_REDUCE_FACTOR_A;
00714         trigger_show_adding_time(-RUN_TIME_LEFT_REDUCE_FACTOR_A);
00715     }
00716
00717     //Si el tiempo restante actual es mayor al mínimo establecido...
00718     else if(new_run_time_left > RUN_TIME_LEFT_REDUCE_UNTIL)
00719     {
00720         new_run_time_left -= RUN_TIME_LEFT_REDUCE_FACTOR_B;
00721         trigger_show_adding_time(-RUN_TIME_LEFT_REDUCE_FACTOR_B);
00722     }
00723
00724 }
00725
00726 static bool is_last_goal(void)
00727 {
00728     int i, j;
00729     for(i = 0, j = 0; i < MAX_GOALS; i++)
00730     {
00731         if(data.goals[i])
00732             j++;
00733     }
00734     //Si es el ultimo goal...
00735     if(j == MAX_GOALS)
00736         return true;
00737     else
00738         return false;
00739 }
```

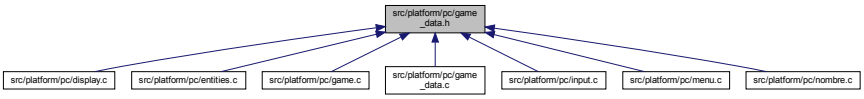
4.33 src/platform/pc/game_data.h File Reference

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 400 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 350 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 358 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 307 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 316 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 447 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 386 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 286 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 381 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 366 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 439 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 422 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 292 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 417 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 465 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 335 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 345 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 302 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 330 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 371 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 227 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 340 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 391 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 432 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 376 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 427 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 325 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 297 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 249 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
```

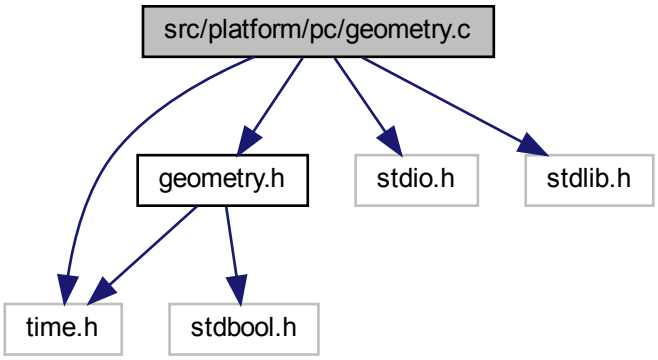
```
00193 bool game_data_get_goal_state(unsigned int goal);
00194
00200 void game_data_set_goal(unsigned int goal);
00201
00206 void game_data_reset_goals(void);
00207
00214 bool game_data_get_game_over_flag(void);
00215
00222 bool game_data_are_goals_full(void);
00223
00229 unsigned long long game_data_get_old_max_score(void);
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 }
```

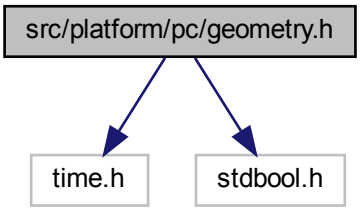
```
00166
00167 pair_xy_t getXYFromTurtleFrame(int frame)
00168 {
00169     return (pair_xy_turtle_sprites_frames[frame]);
00170 }
00171
00172 pair_xy_t getXYFromCarFrame(int frame)
00173 {
00174     return (pair_xy_car_sprites_frames[frame]);
00175 }
00176
00177 pair_xy_t getXYFromCoinFrame(int frame)
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

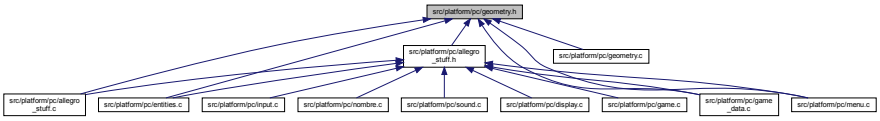
Header del modulo geometry. Defines y enums relacionados a medidas, cantidades y estados para la implementación en PC.

```
#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
```

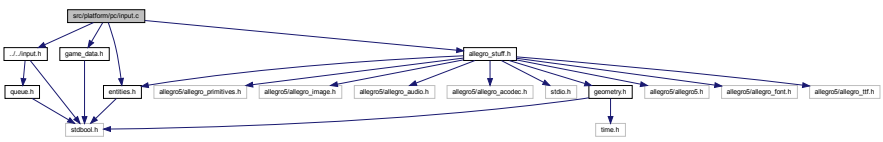
```
00180 typedef enum CAR_TYPE
00181 {
00182     CAR_BLUE = 0,
00183     CAR_POLICE,
00184     CAR_YELLOW,
00185     TRUCK_FIRE,
00186     TRUCK,
00187     CAR_TYPE_N
00188 } CAR_TYPE;
00189
00190 /*****************************************************************************
00191  * VARIABLE PROTOTYPES 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

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

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
00048     if (event != NULL)
00049     {
00050         switch ((*event).type)
00051         {
00052         case ALLEGRO_EVENT_TIMER:
00053             allegro_set_var_redraw(true);
00054             break;
00055
00056         case ALLEGRO_EVENT_KEY_DOWN:
00057             last_key = allegro_get_last_key();
00058
00059             if (last_key != (*event).keyboard.keycode)
00060             {
00061                 retorno = (*event).keyboard.keycode;
00062                 allegro_set_last_key(retorno);
00063
00064                 switch (retorno)
00065                 {
00066                 case ALLEGRO_KEY_F2:
00067                     allegro_sound_set_stream_gain_up();
00068                     break;
00069
00070                 case ALLEGRO_KEY_F1:
00071                     allegro_sound_set_stream_gain_down();
00072                     break;
00073
00074                 case ALLEGRO_KEY_2:
00075                     allegro_sound_unmute();
00076                     break;
00077
00078                 case ALLEGRO_KEY_1:
00079                     allegro_sound_mute();
00080                     break;
00081
00082                 case ALLEGRO_KEY_4:
00083                     game_data_add_score();
00084                     break;
00085
00086                 case ALLEGRO_KEY_5:
00087                     game_data_add_run_time_goal();
00088                     break;
00089
00090                 case ALLEGRO_KEY_6:
00091                     retorno = GAME_OVER;
00092                     break;
00093
00094                 default:
```

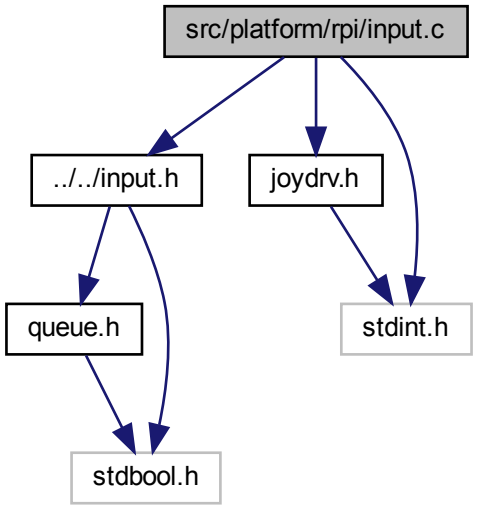
```
00095                     break;
00096                 }
00097             }
00098
00099             break;
00100
00101         case ALLEGRO_EVENT_KEY_UP:
00102             allegro_set_last_key(0);
00103
00104             break;
00105
00106         case ALLEGRO_EVENT_DISPLAY_CLOSE:
00107             retorno = FORCE_SALIR;
00108             break;
00109
00110         default:
00111             break;
00112         }
00113     }
00114
00115     return retorno;
00116 }
```

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)
```

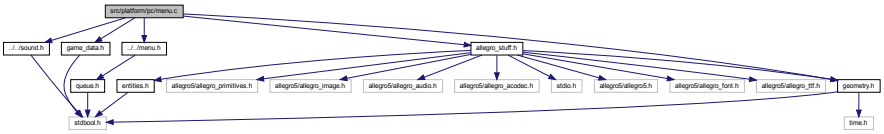
```
00096 {
00097     if (x == -128)
00098         return 127; // excepción: caso en el cual -(-128) = -128
00099
00100     return x >= 0 ? x : -x;
00101 }
```

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);
```

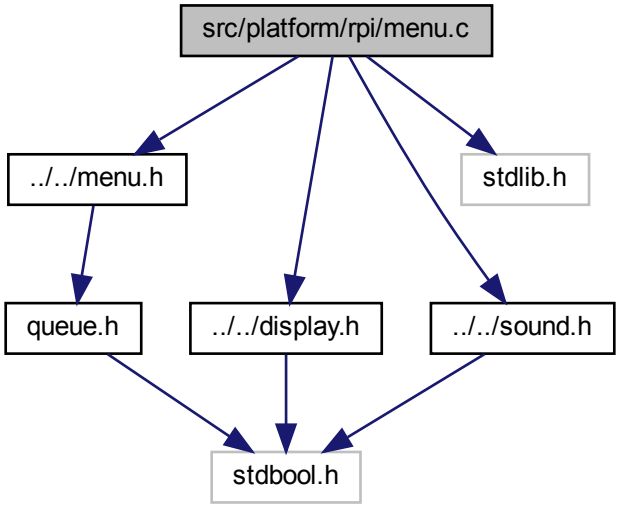
```
00251     }
00252     else if (menu.actual_window == MENU_WINDOW_GAME_OVER)
00253     {
00254         if (score > max_score)
00255         {
00256             al_draw_textf(font, color,
00257                           x,
00258                           y + 10,
00259                           0,
00260                           "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()
```

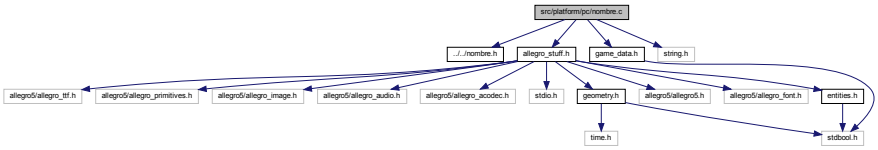
```
00089 {
00090 }
00091
00092 void destruirMenu()
00093 {
00094     free(menu.menu_actual);
00095 }
```

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

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.

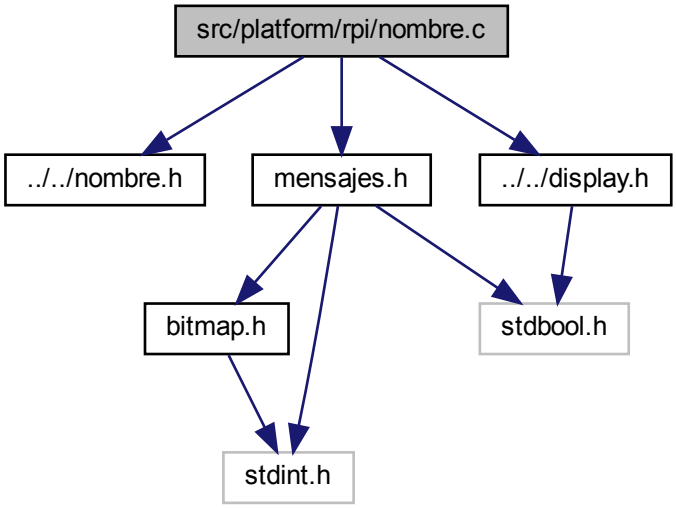
```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "../../nombre.h"
00017
00018 #include "allegro_stuff.h"
00019 #include "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 27 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 /*****************************************************************************
```

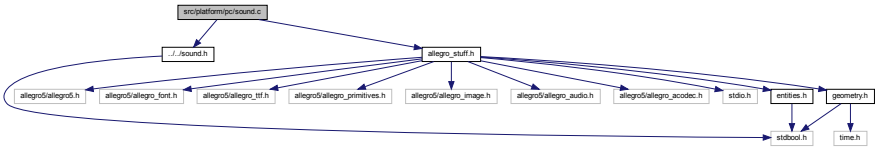
```
00030  ******************************************************************************
00031             GLOBAL FUNCTION DEFINITIONS
00032  ******************************************************************************
00033  ******************************************************************************/
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:
```

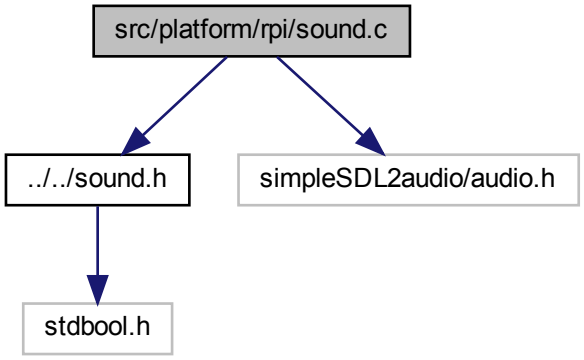
```
00121         break;
00122     }
00123 }
```

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 }
```

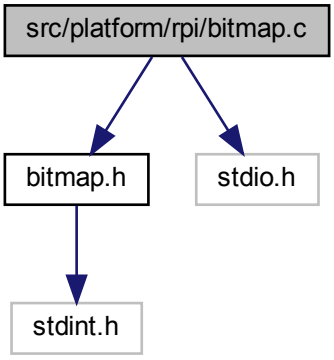
```
00094
00095 void reproducirEfecto(int e)
00096 {
00097     playSound(archivos_efectos[e], SDL_MIX_MAXVOLUME);
00098 }
```

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.

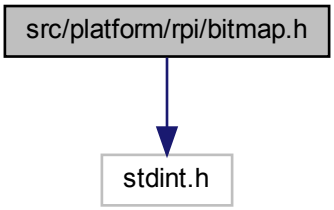
```
00001
00008 /*******************************************************************************
00009  * INCLUDE HEADER FILES
00010  ******************************************************************************/
00011
00012 #include "bitmap.h"
00013
00014 #include <stdio.h>
00015
00016 /*******************************************************************************
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 }
```

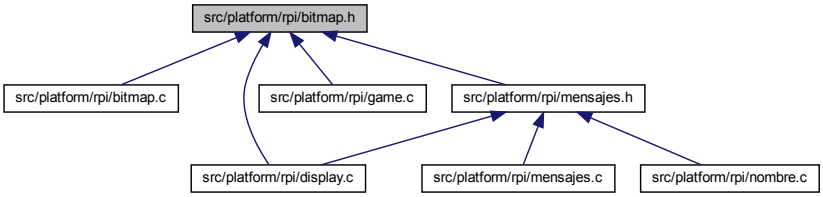
4.57 src/platform/rpi/bitmap.h File Reference

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
```

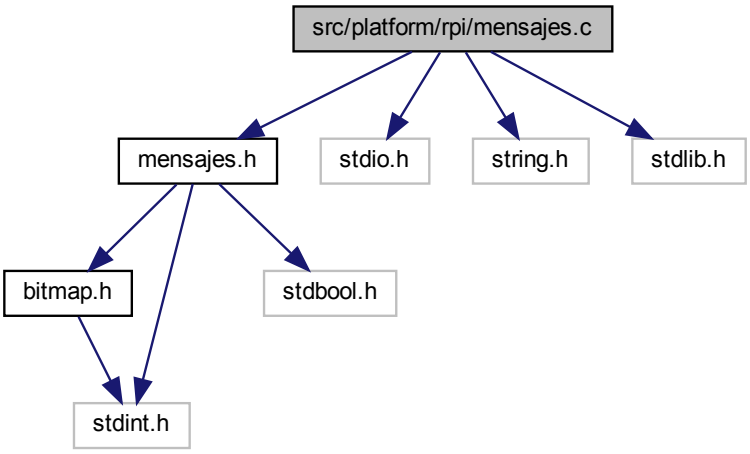
```
00069 jswitch_t joy_get_switch(void);
00070
00071 /*******************************************************************************
00072  ******************************************************************************/
00073
00074 #endif // JOYDRV_H
```

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 110 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 186 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 166 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 304 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 152 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 146 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 232 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 273 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 96 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 220 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 327 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 134 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 158 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 140 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 128 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 116 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 122 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 195 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  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00032  ******************************************************************************/
00033
00040 static int getLongitud(char a);
00041
00042 /*******************************************************************************
00043  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00044  ******************************************************************************/
00045
00046 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, 3, 5,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5}; // sin contar Ñ (+ espacio + numeros + FULL)
00047
00048 /*******************************************************************************
00049  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00050  ******************************************************************************/
00051
00052 static uint16_t char_index[][TAM_RENGLON] = {{0x4000, 0xA000, 0xE000, 0xA000, 0xA000}, // A
00053                                              {0xC000, 0xA000, 0xC000, 0xA000, 0xC000},
00054                                              {0x4000, 0xA000, 0x8000, 0xA000, 0x4000},
00055                                              {0xC000, 0xA000, 0xA000, 0xA000, 0xC000},
00056                                              {0xE000, 0x8000, 0xC000, 0x8000, 0xE000},
00057                                              {0xE000, 0x8000, 0xC000, 0x8000, 0x8000},
00058                                              {0x6000, 0x8000, 0xA000, 0xA000, 0x6000},
00059                                              {0xA000, 0xA000, 0xE000, 0xA000, 0xA000},
00060                                              {0x8000, 0x8000, 0x8000, 0x8000, 0x8000},
00061                                              {0x2000, 0x2000, 0xA000, 0xA000, 0x4000},
```

```
00062                                           {0xA000, 0xA000, 0xC000, 0xA000, 0xA000},
00063                                           {0x8000, 0x8000, 0x8000, 0x8000, 0xE000},
00064                                           {0x8800, 0xD800, 0xA800, 0xA800, 0x8800},
00065                                           {0x9000, 0xD000, 0xB000, 0x9000, 0x9000},
00066                                           {0x4000, 0xA000, 0xA000, 0xA000, 0x4000},
00067                                           {0xC000, 0xA000, 0xA000, 0xC000, 0x8000},
00068                                           {0x4000, 0xA000, 0xA000, 0xA000, 0x6000},
00069                                           {0xC000, 0xA000, 0xC000, 0xA000, 0xA000},
00070                                           {0x6000, 0x8000, 0x4000, 0x2000, 0xC000},
00071                                           {0xE000, 0x4000, 0x4000, 0x4000, 0x4000},
00072                                           {0xA000, 0xA000, 0xA000, 0xA000, 0x4000},
00073                                           {0xA000, 0xA000, 0xA000, 0x4000, 0x4000},
00074                                           {0x8800, 0xA800, 0xA800, 0x5000, 0x5000},
00075                                           {0xA000, 0xA000, 0x4000, 0xA000, 0xA000},
00076                                           {0xA000, 0xA000, 0x4000, 0x4000, 0x4000},
00077                                           {0xE000, 0x2000, 0x4000, 0x8000, 0xE000}, // Z
00078                                           {0x0000, 0x0000, 0x0000, 0x0000, 0x0000}, // espacio
00079                                           {0x4000, 0xA000, 0xA000, 0xA000, 0x4000}, // 0
00080                                           {0x4000, 0xC000, 0x4000, 0x4000, 0xE000},
00081                                           {0x4000, 0xA000, 0x2000, 0x4000, 0xE000},
00082                                           {0xC000, 0x2000, 0x4000, 0x2000, 0xC000},
00083                                           {0xA000, 0xA000, 0xE000, 0x2000, 0x2000},
00084                                           {0xE000, 0x8000, 0xC000, 0x2000, 0xC000},
00085                                           {0x4000, 0x8000, 0xC000, 0xA000, 0xC000},
00086                                           {0xE000, 0x2000, 0x2000, 0x4000, 0x4000},
00087                                           {0x4000, 0xA000, 0x4000, 0xA000, 0x4000},
00088                                           {0x4000, 0xA000, 0x6000, 0x2000, 0x4000},  // 9
00089                                           {0xF800, 0xF800, 0xF800, 0xF800, 0xF800}}; // TODO (FULL)
00090
00091 /*****************************************************************************
00092  *****************************************************************************
00093                           GLOBAL FUNCTION DEFINITIONS
00094  *****************************************************************************
00095  *****************************************************************************/
00096 void printRenglon(renglon_t r)
00097 {
00098     putchar('\n');
00099     for (int i = 0; i < TAM_RENGLON; i++, putchar('\n'))
00100     {
00101         for (int j = 0; j < 2 * CANT_FILAS; j++)
00102         {
00103             putchar((r[i].completo & (0x80000000 >> j)) ? '#' : '-');
00104             if (j == CANT_FILAS - 1)
00105                 putchar('|');
00106         }
00107     }
00108 }
00109
00110 void borrarRenglon(renglon_t r)
00111 {
00112     for (int i = 0; i < TAM_RENGLON; i++)
00113         r[i].completo = 0;
00114 }
00115
00116 void renglonShiftDer(renglon_t r, uint16_t s)
00117 {
00118     for (int i = 0; i < TAM_RENGLON; i++)
00119         r[i].completo >>= s;
00120 }
00121
00122 void renglonShiftIzq(renglon_t r, uint16_t s)
00123 {
00124     for (int i = 0; i < TAM_RENGLON; i++)
00125         r[i].completo <<= s;
00126 }
00127
00128 void renglonOr(renglon_t r, renglon_t s)
00129 {
00130     for (int i = 0; i < TAM_RENGLON; i++)
00131         r[i].completo |= s[i].completo;
00132 }
00133
00134 void renglonAnd(renglon_t r, renglon_t s)
00135 {
00136     for (int i = 0; i < TAM_RENGLON; i++)
00137         r[i].completo &= s[i].completo;
00138 }
00139
00140 void renglonNot(renglon_t r)
00141 {
00142     for (int i = 0; i < TAM_RENGLON; i++)
00143         r[i].completo = ~r[i].completo;
00144 }
00145
00146 void copiarRenglon(renglon_t r1, const renglon_t r2)
00147 {
00148     for (int i = 0; i < TAM_RENGLON; i++)
```

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

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

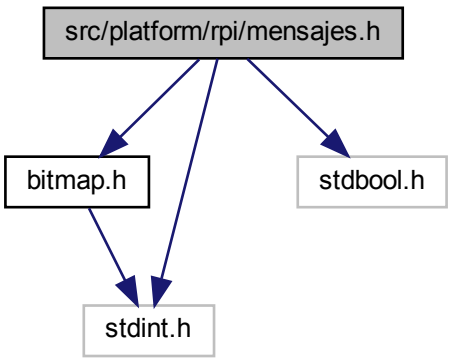
```
00320
00321     renglon_t letra;
00322     charARenglon('A', letra);
00323     renglonShiftDer(letra, msj->j);
00324     renglonOr(msj->renglon, letra);
00325 }
00326
00327 void reemplazarUltLetraMensaje(char c, mensaje_t *msj)
00328 {
00329     (msj->msj)[msj->index] = c;
00330     reemplazarLetra(msj->renglon, c, msj->j);
00331 }
00332
00333 /*******************************************************************************
00334  *******************************************************************************
00335             LOCAL FUNCTION DEFINITIONS
00336  *******************************************************************************
00337  *******************************************************************************/
00338
00339 static int getLongitud(char c)
00340 {
00341     if (c == ' ' || !c)
00342         return longitudes[INDEX_ESPACIO];
00343     else if ('0' <= c && c <= '9')
00344         return longitudes[INDEX_CERO + c - '0'];
00345     else if ('A' <= c && c <= 'Z')
00346         return longitudes[c - 'A'];
00347     else
00348         return longitudes[INDEX_FULL];
00349 }
```

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

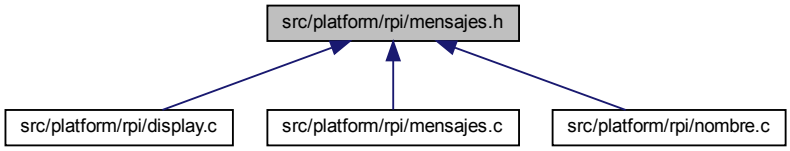
Encabezado de mensajes, con definiciones sobre tipos de datos y funciones.

```
#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 110 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 186 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 166 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 304 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 152 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 146 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 232 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 273 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 96 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 220 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 327 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 134 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 158 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 140 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 128 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 116 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 122 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 195 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  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00059  *****************************************************************************/
00060
00066 void printRenglon(renglon_t r);
00067
00073 void borrarRenglon(renglon_t r);
00074
00081 void renglonShiftDer(renglon_t r, uint16_t s);
00082
00089 void renglonShiftIzq(renglon_t r, uint16_t s);
00090
00097 void renglonOr(renglon_t r, renglon_t s);
00098
00105 void renglonAnd(renglon_t r, renglon_t s);
00106
00112 void renglonNot(renglon_t r);
00113
00120 void copiarRenglon(renglon_t r1, const renglon_t r2);
00121
00129 void copiarMatrizRenglon(matriz_t m, const renglon_t r, int pos);
00130
00138 bool renglonIzquierdoLibre(mensaje_t *msj);
00139
00146 void charARenglon(char c, renglon_t r);
00147
00156 void charAMatriz(char c, matriz_t m, const int coord[]);
00157
00164 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)
00165
00173 void reemplazarLetra(renglon_t r, char c, int j);
00174
```

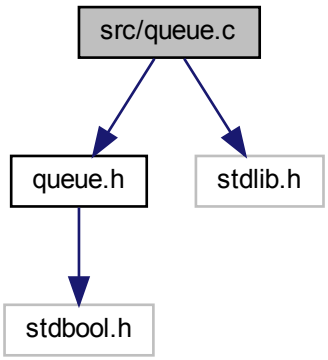
```
00183 mensaje_t mensaje(char *msj, int pos, bool repetir);
00184
00190 void moverMensaje(mensaje_t *msj);
00191
00198 void concatenarLetraMensaje(char c, mensaje_t *msj);
00199
00206 void reemplazarUltLetraMensaje(char c, mensaje_t *msj);
00207
00208 /*****************************************************************************
00209  *****************************************************************************/
00210
00211 #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.

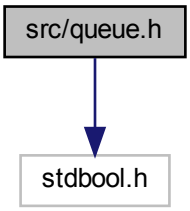
```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "queue.h"
00017
00018 #include <stdlib.h>
00019
00020 /*******************************************************************************
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 }
```

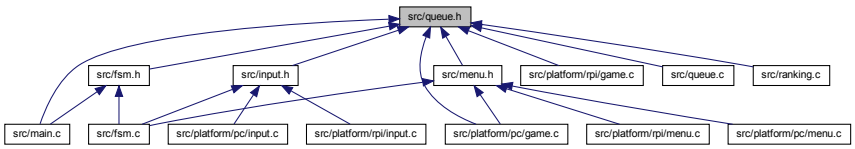
4.67 src/queue.h File Reference

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** = -2 }
- 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 = -2
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,
```

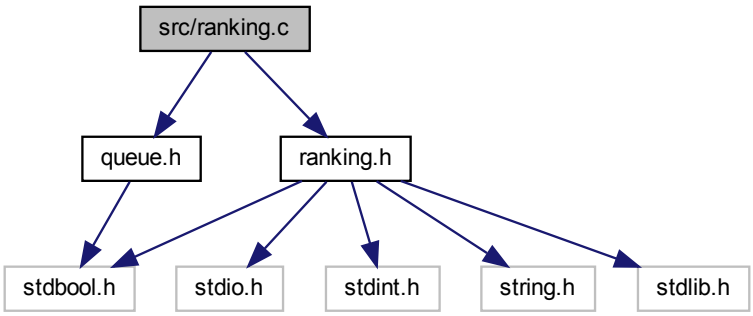
```
00049     ABAJO
00050 };
00051
00052 /*******************************************************************************
00053  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00054  ******************************************************************************/
00055
00060 void queueInsertar(event_t);
00061
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

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

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 174 of file ranking.c.

4.69.3.4 getRankingLineas() `uint getRankingLineas ( void )`

Devuelve cantidad de renglones del ranking.

Returns

int Renglones

Definition at line 194 of file ranking.c.

4.69.3.5 getRankingNombres() `char ** getRankingNombres ( void )`

Devuelve array de nombres de jugadores.

Returns

char**

Definition at line 199 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 204 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 160 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) + 1);
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     free(scores);
00158 }
00159
00160 bool verificarJugadorRanking(char *name)
00161 {
00162     // Ranking vacio
00163     if (!lineNumber)
00164         return false;
00165
00166     int i;
00167     bool exists;
00168     for (i = 0, exists = false; i < lineNumber && !exists; i++)
00169         exists = strcmp(names[i], name) == 0;
00170
00171     return exists;
00172 }
00173
00174 unsigned long long getJugadorRankingPuntos(char *name)
00175 {
```

```
00176     int i;
00177     bool exists;
00178     unsigned long long score = 0;
00179 
00180     for (i = 0, exists = false; i < lineNumber && !exists; i++)
00181     {
00182         // Si el nombre coincide...
00183         if (strcmp(names[i], name) == 0)
00184         {
00185             // Carga el score
00186             score = scores[i];
00187             exists = true;
00188         }
00189     }
00190 
00191     return score;
00192 }
00193 
00194 uint getRankingLineas(void)
00195 {
00196     return lineNumber;
00197 }
00198 
00199 char **getRankingNombres(void)
00200 {
00201     return names;
00202 }
00203 
00204 unsigned long long *getRankingPuntos(void)
00205 {
00206     return scores;
00207 }
00208 
00209 /*****************************************************************************
00210  *****************************************************************************
00211                         LOCAL FUNCTION DEFINITIONS
00212  *****************************************************************************
00213  *****************************************************************************/
00214 
00215 static void recargarRanking(void)
00216 {
00217     lineNumber = 0;
00218 
00219     while (fgets(tempStr, MAX_LEN, handlerRanking) != NULL)
00220     {
00221         char *p = strchr(tempStr, '\n');
00222         while (p != NULL)
00223         {
00224             *p = '\0'; // Saco todos los saltos de linea
00225             p = strchr(tempStr, '\n');
00226         }
00227 
00228         char *tempPtr = strtok(tempStr, " ");                                   // Apunto al
      nombre
00229         names = (char **)realocar(names, sizeof(char *) * (lineNumber + 1));        // Reservo memoria
      para un puntero
00230         names[lineNumber] = (char *)realocar(NULL, strlen(tempPtr) * sizeof(char) + 1); // Reservo
      memoria para el nombre
00231         strcpy(names[lineNumber], tempPtr);
00232 
00233         tempPtr = strtok(NULL, " ");
       // Apunto a los puntos
00234         scores = (unsigned long long *)realocar(scores, sizeof(unsigned long long) * (lineNumber +
      1)); // Reservo memoria para un score
00235         scores[lineNumber] = strtoul(tempPtr, NULL, 10);
00236 
00237         lineNumber++;
00238     }
00239 }
00240 
00241 static void ordenarRanking(void)
00242 {
00243     int i, j;
00244     unsigned long long tempScore;
00245 
00246     for (i = 0; i < (lineNumber - 1); i++)
00247     {
00248         for (j = 0; j < (lineNumber - i - 1); j++)
00249         {
00250             // Si el primer score es menor, o si es igual al siguiente pero predomina orden
      alfabetico...
00251             if ((scores[j] < scores[j + 1]) || ((scores[j] == scores[j + 1]) && (strcmp(names[j],
      names[j + 1]) > 0)))
00252             {
00253                 // Backup del menor
00254                 strcpy(tempStr, names[j]);
00255                 tempScore = scores[j];
```

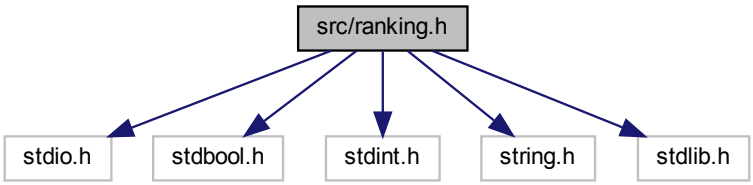
```
00256
00257                 // El mayor se pone en la posicion del menor
00258                 strcpy(names[j], names[j + 1]);
00259                 scores[j] = scores[j + 1];
00260
00261                 // El backup se pone en la posicion del mayor
00262                 strcpy(names[j + 1], tempStr);
00263                 scores[j + 1] = tempScore;
00264             }
00265         }
00266     }
00267 }
00268
00269 static void writeRanking(void)
00270 {
00271     int i;
00272
00273     // Crea archivo temporal
00274     if ((handlerTemp = fopen(strTemp, "w")) == NULL)
00275         printf("Error opening temp.txt");
00276
00277     if (lineNumber)
00278     {
00279         // Copia lo nuevo en temp.txt
00280         for (i = 0; i < lineNumber; i++)
00281             fprintf(handlerTemp, "%s %lld\n", names[i], scores[i]);
00282     }
00283
00284     remove(strRanking);
00285     rename(strTemp, strRanking);
00286
00287     fclose(handlerTemp);
00288 }
00289
00290 static void createRankingFile(void)
00291 {
00292     // crea el archivo, si no lo estaba
00293     FILE *pFile;
00294     if ((pFile = fopen(strRanking, "a")) == NULL)
00295     {
00296         printf("Error creando %s", strRanking);
00297     }
00298     fclose(pFile);
00299 }
00300
00301 static void *realocar(void *p, size_t n)
00302 {
00303     void *aux = realloc(p, n);
00304     if (aux == NULL)
00305     {
00306         perror("Error en ranking.c al realocar memoria\n");
00307         free(p);
00308         queueInsertar(FORCE_SALIR);
00309     }
00310     return aux;
00311 }
```

4.71 src/ranking.h File Reference

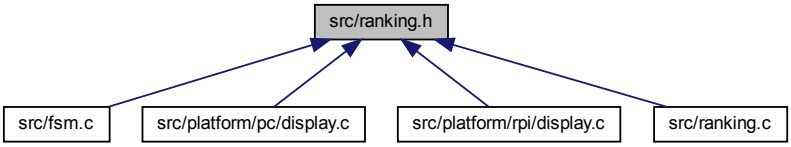
Header del modulo ranking. Prototipos de funciones de interaccion con el ranking de jugadores. Permite trabajar con el txt correspondiente fácilmente,.

```
#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 174 of file ranking.c.

4.71.3.4 getRankingLineas() `uint getRankingLineas ( void )`

Devuelve cantidad de renglones del ranking.

Returns

int Renglones

Definition at line 194 of file ranking.c.

4.71.3.5 getRankingNombres() `char ** getRankingNombres ( void )`

Devuelve array de nombres de jugadores.

Returns

char∗∗

Definition at line 199 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 204 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 160 of file ranking.c.

4.72 ranking.h

Go to the documentation of this file.

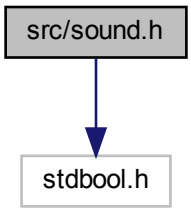
```
00001
00013 #ifndef _RANKING_H_
00014 #define _RANKING_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdio.h>
00021 #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_
```

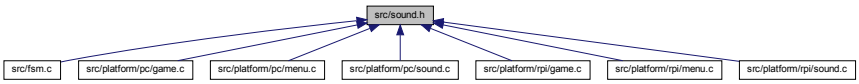
4.73 src/sound.h File Reference

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