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

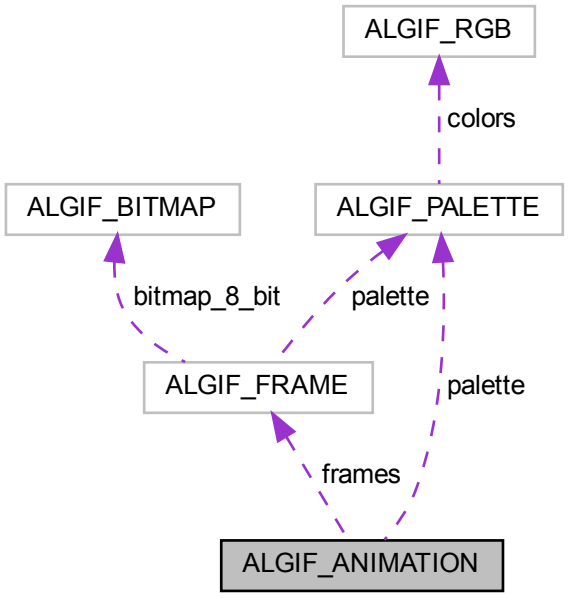
2.1 File List

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

3 Data Structure Documentation

3.1 ALGIF_ANIMATION Struct Reference

Collaboration diagram for ALGIF_ANIMATION:



Data Fields

- int width
- int height
- int frames_count
- int background_index
- int loop
- ALGIF_PALETTE palette
- ALGIF_FRAME * frames
- int duration
- ALLEGRO_BITMAP * store

3.1.1 Detailed Description

Definition at line 43 of file algif.h.

3.1.2 Field Documentation

3.1.2.1 background_index `int background_index`

Definition at line 47 of file algif.h.

3.1.2.2 duration `int duration`

Definition at line 52 of file algif.h.

3.1.2.3 frames `ALGIF_FRAME* frames`

Definition at line 50 of file algif.h.

3.1.2.4 frames_count `int frames_count`

Definition at line 46 of file algif.h.

3.1.2.5 height `int height`

Definition at line 45 of file algif.h.

3.1.2.6 loop `int loop`

Definition at line 48 of file algif.h.

3.1.2.7 palette `ALGIF_PALETTE palette`

Definition at line 49 of file algif.h.

3.1.2.8 store `ALLEGRO_BITMAP* store`

Definition at line 53 of file algif.h.

3.1.2.9 width `int width`

Definition at line 45 of file algif.h.

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

- src/platform/pc/algif5/algif.h

3.2 ALGIF_BITMAP Struct Reference

Data Fields

- int w
- int h
- uint8_t * data

3.2.1 Detailed Description

Definition at line 37 of file algif.h.

3.2.2 Field Documentation

3.2.2.1 data `uint8_t* data`

Definition at line 40 of file algif.h.

3.2.2.2 h `int h`

Definition at line 39 of file algif.h.

3.2.2.3 w `int w`

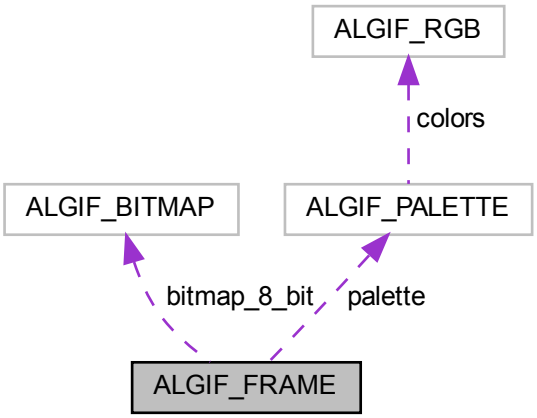
Definition at line 39 of file algif.h.

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

- src/platform/pc/algif5/algif.h

3.3 ALGIF_FRAME Struct Reference

Collaboration diagram for ALGIF_FRAME:



Data Fields

- ALGIF_BITMAP * bitmap_8_bit
- ALGIF_PALETTE palette
- int xoff
- int yoff
- int duration
- int disposal_method
- int transparent_index
- ALLEGRO_BITMAP * rendered

3.3.1 Detailed Description

Definition at line 56 of file algif.h.

3.3.2 Field Documentation

3.3.2.1 bitmap_8_bit `ALGIF_BITMAP* bitmap_8_bit`

Definition at line 58 of file algif.h.

3.3.2.2 disposal_method `int disposal_method`

Definition at line 62 of file algif.h.

3.3.2.3 duration `int duration`

Definition at line 61 of file algif.h.

3.3.2.4 palette `ALGIF_PALETTE palette`

Definition at line 59 of file algif.h.

3.3.2.5 rendered `ALLEGRO_BITMAP* rendered`

Definition at line 65 of file algif.h.

3.3.2.6 transparent_index `int transparent_index`

Definition at line 63 of file algif.h.

3.3.2.7 xoff `int xoff`

Definition at line 60 of file algif.h.

3.3.2.8 yoff `int yoff`

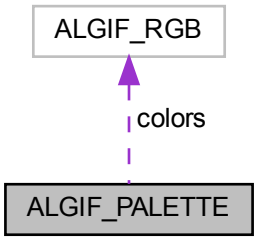
Definition at line 60 of file algif.h.

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

- src/platform/pc/algif5/algif.h

3.4 ALGIF_PALETTE Struct Reference

Collaboration diagram for ALGIF_PALETTE:



Data Fields

- int colors_count
- ALGIF_RGB colors [256]

3.4.1 Detailed Description

Definition at line 31 of file algif.h.

3.4.2 Field Documentation

3.4.2.1 colors `ALGIF_RGB colors[256]`

Definition at line 34 of file algif.h.

3.4.2.2 colors_count `int colors_count`

Definition at line 33 of file algif.h.

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

- src/platform/pc/algif5/algif.h

3.5 ALGIF_RGB Struct Reference

Data Fields

- uint8_t r
- uint8_t g
- uint8_t b

3.5.1 Detailed Description

Definition at line 26 of file algif.h.

3.5.2 Field Documentation

3.5.2.1 b `uint8_t b`

Definition at line 28 of file algif.h.

3.5.2.2 g `uint8_t g`

Definition at line 28 of file algif.h.

3.5.2.3 r `uint8_t r`

Definition at line 28 of file algif.h.

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

- src/platform/pc/algif5/algif.h

3.6 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

3.6.1 Detailed Description

Definition at line 87 of file allegro_stuff.c.

3.6.2 Field Documentation

3.6.2.1 disp `ALLEGRO_DISPLAY* disp`

Definition at line 96 of file allegro_stuff.c.

3.6.2.2 done `bool done`

Definition at line 107 of file allegro_stuff.c.

3.6.2.3 event `ALLEGRO_EVENT event`

Definition at line 104 of file allegro_stuff.c.

3.6.2.4 font `ALLEGRO_FONT* font`

Definition at line 99 of file allegro_stuff.c.

3.6.2.5 font_h `int font_h`

Definition at line 100 of file allegro_stuff.c.

3.6.2.6 font_w `int font_w`

Definition at line 101 of file allegro_stuff.c.

3.6.2.7 queue `ALLEGRO_EVENT_QUEUE* queue`

Definition at line 93 of file allegro_stuff.c.

3.6.2.8 redraw `bool redraw`

Definition at line 109 of file allegro_stuff.c.

3.6.2.9 timer `ALLEGRO_TIMER* timer`

Definition at line 90 of file allegro_stuff.c.

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

- src/platform/pc/allegro_stuff.c

3.7 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

3.7.1 Detailed Description

Definition at line 78 of file entities.c.

3.7.2 Field Documentation

3.7.2.1 count `int count`

Definition at line 86 of file entities.c.

3.7.2.2 dx `int dx`

Definition at line 83 of file entities.c.

3.7.2.3 fast `bool fast`

Definition at line 87 of file entities.c.

3.7.2.4 lane `int lane`

Definition at line 82 of file entities.c.

3.7.2.5 length `int length`

Definition at line 85 of file entities.c.

3.7.2.6 type `CAR_TYPE type`

Definition at line 84 of file entities.c.

3.7.2.7 used `bool used`

Definition at line 88 of file entities.c.

3.7.2.8 x `int x`

Definition at line 80 of file entities.c.

3.7.2.9 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

3.8 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**

3.8.1 Detailed Description

Definition at line 121 of file entities.c.

3.8.2 Field Documentation

3.8.2.1 blink_timer `unsigned int blink_timer`

Definition at line 130 of file entities.c.

3.8.2.2 cont `unsigned int cont`

Definition at line 131 of file entities.c.

3.8.2.3 flag `bool flag`

Definition at line 132 of file entities.c.

3.8.2.4 frame_cont `unsigned int frame_cont`

Definition at line 128 of file entities.c.

3.8.2.5 timeout `unsigned int timeout`

Definition at line 129 of file entities.c.

3.8.2.6 used `bool used`

Definition at line 125 of file entities.c.

3.8.2.7 x `int x`

Definition at line 123 of file entities.c.

3.8.2.8 y `int y`

Definition at line 124 of file entities.c.

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

- src/platform/pc/entities.c

3.9 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]

3.9.1 Detailed Description

Definition at line 43 of file game_data.c.

3.9.2 Field Documentation

3.9.2.1 difficulty `int difficulty`

Definition at line 60 of file game_data.c.

3.9.2.2 flag `unsigned char flag`

Definition at line 64 of file game_data.c.

3.9.2.3 frames `unsigned long frames`

Definition at line 57 of file game_data.c.

3.9.2.4 goals `bool goals[MAX_GOALS]`

Definition at line 66 of file game_data.c.

3.9.2.5 lives `int lives`

Definition at line 45 of file game_data.c.

3.9.2.6 name `char name[MAX_NAME_CHAR]`

Definition at line 62 of file game_data.c.

3.9.2.7 number `int number`

Definition at line 51 of file game_data.c.

3.9.2.8 score `unsigned long long score`

Definition at line 46 of file game_data.c.

3.9.2.9 score_max `unsigned long long score_max`

Definition at line 47 of file game_data.c.

3.9.2.10 time `int time`

Definition at line 53 of file game_data.c.

3.9.2.11 time_left `int time_left`

Definition at line 52 of file game_data.c.

3.9.2.12 time_ref `long time_ref`

Definition at line 54 of file game_data.c.

3.9.2.13 timer_in_sec `int timer_in_sec`

Definition at line 58 of file game_data.c.

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

- src/platform/pc/game_data.c

3.10 dcoord_t Struct Reference

Data Fields

- uint8_t x
- uint8_t y

3.10.1 Detailed Description

Definition at line 36 of file disdrv.h.

3.10.2 Field Documentation

3.10.2.1 x `uint8_t x`

Definition at line 38 of file disdrv.h.

3.10.2.2 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

3.11 frog_t Struct Reference

Data Fields

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

3.11.1 Detailed Description

Definition at line 66 of file entities.c.

3.11.2 Field Documentation

3.11.2.1 facing `int facing`

Definition at line 71 of file entities.c.

3.11.2.2 moving `int moving`

Definition at line 70 of file entities.c.

3.11.2.3 next_action `unsigned char next_action`

Definition at line 74 of file entities.c.

3.11.2.4 state `unsigned char state`

Definition at line 73 of file entities.c.

3.11.2.5 steps `int steps`

Definition at line 72 of file entities.c.

3.11.2.6 x `int x`

Definition at line 68 of file entities.c.

3.11.2.7 y `int y`

Definition at line 69 of file entities.c.

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

- src/platform/pc/entities.c

3.12 jcoord_t Struct Reference

Data Fields

- int8_t x
- int8_t y

3.12.1 Detailed Description

Definition at line 33 of file joydrv.h.

3.12.2 Field Documentation

3.12.2.1 x `int8_t x`

Definition at line 35 of file joydrv.h.

3.12.2.2 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

3.13 log_t Struct Reference

Data Fields

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

3.13.1 Detailed Description

Definition at line 91 of file entities.c.

3.13.2 Field Documentation

3.13.2.1 dx `int dx`

Definition at line 96 of file entities.c.

3.13.2.2 lane `int lane`

Definition at line 95 of file entities.c.

3.13.2.3 used `bool used`

Definition at line 97 of file entities.c.

3.13.2.4 x `int x`

Definition at line 93 of file entities.c.

3.13.2.5 y `int y`

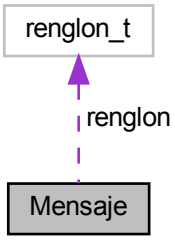
Definition at line 94 of file entities.c.

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

- src/platform/pc/entities.c

3.14 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

3.14.1 Detailed Description

Definition at line 44 of file mensajes.h.

3.14.2 Field Documentation

3.14.2.1 habilitacion `bool habilitacion`

Definition at line 51 of file mensajes.h.

3.14.2.2 index `int index`

Definition at line 48 of file mensajes.h.

3.14.2.3 j `int j`

Definition at line 50 of file mensajes.h.

3.14.2.4 longitud `int longitud`

Definition at line 49 of file mensajes.h.

3.14.2.5 mover_texto `bool mover_texto`

Definition at line 52 of file mensajes.h.

3.14.2.6 msj `char msj[L_MAX]`

Definition at line 46 of file mensajes.h.

3.14.2.7 posicion `int posicion`

Definition at line 47 of file mensajes.h.

3.14.2.8 renglon `renglon_t renglon`

Definition at line 54 of file mensajes.h.

3.14.2.9 repetir_msj `bool repetir_msj`

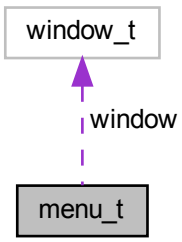
Definition at line 53 of file mensajes.h.

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

- src/platform/rpi/mensajes.h

3.15 menu_t Struct Reference

Collaboration diagram for menu_t:



Data Fields

- window_t window [MENU_WINDOW_MAX]
- int actual_window

3.15.1 Detailed Description

Definition at line 40 of file menu.c.

3.15.2 Field Documentation

3.15.2.1 actual_window `int actual_window`

Definition at line 44 of file menu.c.

3.15.2.2 window `window_t window[MENU_WINDOW_MAX]`

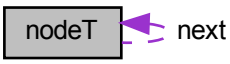
Definition at line 42 of file menu.c.

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

- src/platform/pc/menu.c

3.16 nodeT Struct Reference

Collaboration diagram for nodeT:



Data Fields

- event_t data
- struct nodeT * next

3.16.1 Detailed Description

Definition at line 27 of file queue.c.

3.16.2 Field Documentation

3.16.2.1 data `event_t data`

Definition at line 29 of file queue.c.

3.16.2.2 next `struct nodeT* next`

Definition at line 30 of file queue.c.

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

- src/queue.c

3.17 pair_xy_t Struct Reference

Data Fields

- int x
- int y

3.17.1 Detailed Description

Definition at line 135 of file geometry.h.

3.17.2 Field Documentation

3.17.2.1 x `int x`

Definition at line 137 of file geometry.h.

3.17.2.2 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

3.18 privateAudioDevice Struct Reference

Data Fields

- SDL_AudioDeviceID device
- SDL_AudioSpec want
- uint8_t audioEnabled

3.18.1 Detailed Description

Definition at line 71 of file audio.c.

3.18.2 Field Documentation

3.18.2.1 audioEnabled `uint8_t audioEnabled`

Definition at line 75 of file audio.c.

3.18.2.2 device `SDL_AudioDeviceID device`

Definition at line 73 of file audio.c.

3.18.2.3 want `SDL_AudioSpec want`

Definition at line 74 of file audio.c.

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

- src/platform/rpi/simpleSDL2audio/audio.c

3.19 renglon_t Union Reference

Data Fields

- uint32_t completo
-

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

3.19.1 Detailed Description

Definition at line 34 of file mensajes.h.

3.19.2 Field Documentation

3.19.2.1 completo `uint32_t completo`

Definition at line 36 of file mensajes.h.

3.19.2.2 mitad_der `uint16_t mitad_der`

Definition at line 39 of file mensajes.h.

3.19.2.3 mitad_izq `uint16_t mitad_izq`

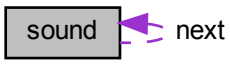
Definition at line 40 of file mensajes.h.

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

- src/platform/rpi/mensajes.h

3.20 sound Struct Reference

Collaboration diagram for sound:



Data Fields

- uint32_t length
- uint32_t lengthTrue
- uint8_t * bufferTrue
- uint8_t * buffer
- uint8_t loop
- uint8_t fade
- uint8_t free
- uint8_t volume
- SDL_AudioSpec audio
- struct sound * next

3.20.1 Detailed Description

Definition at line 40 of file audio.h.

3.20.2 Field Documentation

3.20.2.1 audio `SDL_AudioSpec audio`

Definition at line 51 of file audio.h.

3.20.2.2 buffer `uint8_t* buffer`

Definition at line 45 of file audio.h.

3.20.2.3 bufferTrue `uint8_t* bufferTrue`

Definition at line 44 of file audio.h.

3.20.2.4 fade `uint8_t fade`

Definition at line 47 of file audio.h.

3.20.2.5 free `uint8_t free`

Definition at line 48 of file audio.h.

3.20.2.6 length `uint32_t length`

Definition at line 42 of file audio.h.

3.20.2.7 lengthTrue `uint32_t lengthTrue`

Definition at line 43 of file audio.h.

3.20.2.8 loop `uint8_t loop`

Definition at line 46 of file audio.h.

3.20.2.9 next `struct sound* next`

Definition at line 53 of file audio.h.

3.20.2.10 volume `uint8_t volume`

Definition at line 49 of file audio.h.

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

- src/platform/rpi/simpleSDL2audio/audio.h

3.21 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**

3.21.1 Detailed Description

Definition at line 113 of file allegro_stuff.c.

3.21.2 Field Documentation

3.21.2.1 bonus `ALLEGRO_SAMPLE* bonus`

Definition at line 125 of file allegro_stuff.c.

3.21.2.2 click `ALLEGRO_SAMPLE* click`

Definition at line 124 of file allegro_stuff.c.

3.21.2.3 coin_drop `ALLEGRO_SAMPLE* coin_drop`

Definition at line 132 of file allegro_stuff.c.

3.21.2.4 crash `ALLEGRO_SAMPLE* crash`

Definition at line 121 of file allegro_stuff.c.

3.21.2.5 drowned `ALLEGRO_SAMPLE* drowned`

Definition at line 127 of file allegro_stuff.c.

3.21.2.6 exiting `ALLEGRO_SAMPLE* exiting`

Definition at line 130 of file allegro_stuff.c.

3.21.2.7 goal `ALLEGRO_SAMPLE* goal`

Definition at line 122 of file allegro_stuff.c.

3.21.2.8 jump `ALLEGRO_SAMPLE* jump`

Definition at line 120 of file allegro_stuff.c.

3.21.2.9 low_time `ALLEGRO_SAMPLE* low_time`

Definition at line 123 of file allegro_stuff.c.

3.21.2.10 menu_enter `ALLEGRO_SAMPLE* menu_enter`

Definition at line 128 of file allegro_stuff.c.

3.21.2.11 new_max_score `ALLEGRO_SAMPLE* new_max_score`

Definition at line 129 of file allegro_stuff.c.

3.21.2.12 no_time `ALLEGRO_SAMPLE* no_time`

Definition at line 131 of file allegro_stuff.c.

3.21.2.13 run_completed `ALLEGRO_SAMPLE* run_completed`

Definition at line 126 of file allegro_stuff.c.

3.21.2.14 stream `ALLEGRO_AUDIO_STREAM* stream`

Definition at line 115 of file allegro_stuff.c.

3.21.2.15 stream_state `unsigned char stream_state`

Definition at line 116 of file allegro_stuff.c.

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

- src/platform/pc/allegro_stuff.c

3.22 sprites_menu_t Struct Reference

3.22.1 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

3.23 sprites_t Struct Reference

Estructura principal de spritesheets.

`#include <allegro_stuff.h>`

Data Fields

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

3.23.1 Detailed Description

Estructura principal de spritesheets.

Definition at line 59 of file allegro_stuff.h.

3.23.2 Field Documentation

3.23.2.1 background `ALLEGRO_BITMAP* background`

Definition at line 64 of file allegro_stuff.h.

3.23.2.2 border `ALLEGRO_BITMAP* border`

Definition at line 103 of file allegro_stuff.h.

3.23.2.3 car `ALLEGRO_BITMAP* car[CAR_TYPE_N]`

Definition at line 69 of file allegro_stuff.h.

3.23.2.4 cars_uncut `ALLEGRO_BITMAP* cars_uncut`

Definition at line 68 of file allegro_stuff.h.

3.23.2.5 credits `ALLEGRO_BITMAP* credits`

Definition at line 83 of file allegro_stuff.h.

3.23.2.6 dead `ALLEGRO_BITMAP* dead`

Definition at line 89 of file allegro_stuff.h.

3.23.2.7 frame ALLEGRO_BITMAP* frame[SPRITE_SPLASH_FRAMES]

Definition at line 94 of file allegro_stuff.h.

3.23.2.8 frog ALLEGRO_BITMAP* frog[8]

Definition at line 62 of file allegro_stuff.h.

3.23.2.9 frog_uncut ALLEGRO_BITMAP* frog_uncut

Definition at line 61 of file allegro_stuff.h.

3.23.2.10 heart ALLEGRO_BITMAP* heart

Definition at line 74 of file allegro_stuff.h.

3.23.2.11 icon ALLEGRO_BITMAP* icon

Definition at line 87 of file allegro_stuff.h.

3.23.2.12 log ALLEGRO_BITMAP* log

Definition at line 66 of file allegro_stuff.h.

3.23.2.13 name ALLEGRO_BITMAP* name

Definition at line 85 of file allegro_stuff.h.

3.23.2.14 option ALLEGRO_BITMAP* option[MENU_STATE_MAX]

Definition at line 79 of file allegro_stuff.h.

3.23.2.15 turtle ALLEGRO_BITMAP* turtle[TURTLE_FRAMES]

Definition at line 72 of file allegro_stuff.h.

3.23.2.16 turtle_uncut ALLEGRO_BITMAP* turtle_uncut

Definition at line 71 of file allegro_stuff.h.

3.23.2.17 uncut ALLEGRO_BITMAP* uncut

Definition at line 78 of file allegro_stuff.h.

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

- src/platform/pc/allegro_stuff.h

3.24 state_diagram_edge Struct Reference

Collaboration diagram for state_diagram_edge:

Data Fields

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

3.24.1 Detailed Description

Definition at line 53 of file fsm.c.

3.24.2 Field Documentation

3.24.2.1 evento `event_t evento`

Definition at line 55 of file fsm.c.

3.24.2.2 p_rut_accion `void(* p_rut_accion) (void)`

Definition at line 57 of file fsm.c.

3.24.2.3 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

3.25 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

3.25.1 Detailed Description

Definition at line 101 of file entities.c.

3.25.2 Field Documentation

3.25.2.1 cont `unsigned int cont`

Definition at line 114 of file entities.c.

3.25.2.2 dx `int dx`

Definition at line 106 of file entities.c.

3.25.2.3 frame `unsigned char frame`

Definition at line 112 of file entities.c.

3.25.2.4 lane `int lane`

Definition at line 105 of file entities.c.

3.25.2.5 state `unsigned char state`

Definition at line 118 of file entities.c.

3.25.2.6 timeout `unsigned int timeout`

Definition at line 113 of file entities.c.

3.25.2.7 turtles_in_pack `unsigned char turtles_in_pack`

Definition at line 108 of file entities.c.

3.25.2.8 used `bool used`

Definition at line 107 of file entities.c.

3.25.2.9 wide `int wide`

Definition at line 117 of file entities.c.

3.25.2.10 x `int x`

Definition at line 103 of file entities.c.

3.25.2.11 y `int y`

Definition at line 104 of file entities.c.

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

- src/platform/pc/entities.c

3.26 window_t Struct Reference

Data Fields

- int actual_state
- int max_states

3.26.1 Detailed Description

Definition at line 34 of file menu.c.

3.26.2 Field Documentation

3.26.2.1 actual_state `int actual_state`

Definition at line 36 of file menu.c.

3.26.2.2 max_states `int max_states`

Definition at line 37 of file menu.c.

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

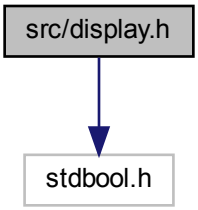
- src/platform/pc/menu.c

4 File Documentation

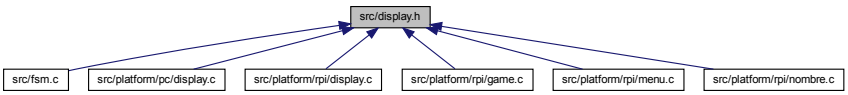
4.1 src/display.h File Reference

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

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



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



Enumerations

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

Functions

- bool iniciarDisplay ()

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

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

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

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

Deja el texto en la posición data (no retiene)

- void cargarRanking (void)

  Inicia muestreo de ranking en la plataforma.

- void mostrarRanking (void)

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

- void cargarCreditos (void)

  Inicializa los cretidos en la plataforma.

- void mostrarCreditos (void)

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

- void reconfigurarDisplayON (void)

  Reconfigura el display de la plataforma y lo habilita.

- void reconfigurarDisplayOFF (void)

  Reconfigura el display de la plataforma y lo deshabilita.

4.1.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file display.h.

4.1.2 Enumeration Type Documentation

4.1.2.1 posiciones_mensajes `enum posiciones_mensajes`

Definition at line 27 of file display.h.

4.1.3 Function Documentation

4.1.3.1 actualizarDisplay() `void actualizarDisplay ( )`

Actualiza el display de la plataforma.

Definition at line 58 of file display.c.

4.1.3.2 cargarCreditos() `void cargarCreditos ( void )`

Inicializa los cretidos en la plataforma.

Definition at line 140 of file display.c.

4.1.3.3 cargarRanking() `void cargarRanking ( void )`

Inicia muestreo de ranking en la plataforma.

Parameters

txt	

Definition at line 78 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 73 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 63 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 145 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 135 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 68 of file display.c.

4.1.3.10 reconfigurarDisplayOFF() `void reconfigurarDisplayOFF ( void )`

Reconfigura el display de la plataforma y lo deshabilita.

Definition at line 167 of file display.c.

4.1.3.11 reconfigurarDisplayON() void reconfigurarDisplayON (void)

Reconfigura el display de la plataforma y lo habilita.

Definition at line 162 of file display.c.

4.2 display.h

Go to the documentation of this file.

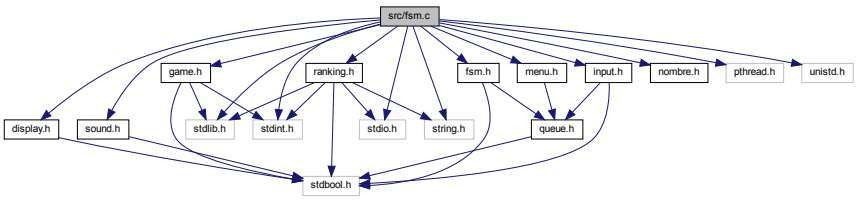
```
00001
00013 #ifndef _DISPLAY_H_
00014 #define _DISPLAY_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdbool.h>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

4.3 src/fsm.c File Reference

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

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

Include dependency graph for fsm.c:



Data Structures

- struct state_diagram_edge

Macros

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

Typedefs

- typedef struct state_diagram_edge STATE

Functions

- bool inicializarFsm (void)

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

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

  Fixea consumo elevado de cpu en el while loop principal.

Variables

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

4.3.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file fsm.c.

4.3.2 Macro Definition Documentation

4.3.2.1 CTE_OPCION `#define CTE_OPCION 100`

Definition at line 41 of file fsm.c.

4.3.2.2 FIN_TABLA `#define FIN_TABLA 0xFF`

Definition at line 38 of file fsm.c.

4.3.2.3 FIX_CPU_USAGE_SLEEP_US `#define FIX_CPU_USAGE_SLEEP_US 500`

Definition at line 44 of file fsm.c.

4.3.3 Typedef Documentation

4.3.3.1 STATE `typedef struct state_diagram_edge STATE`

Definition at line 50 of file fsm.c.

4.3.4 Function Documentation

4.3.4.1 fixHighCpuUsage() `void fixHighCpuUsage ( void )`

Fixea consumo elevado de cpu en el while loop principal.

Not the best solucion, but sirve...

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

Definition at line 356 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 338 of file fsm.c.

4.3.4.3 inicializarFsm() `bool inicializarFsm ( void )`

Inicializa la FSM, notificando si tuvo exito.

Returns

true Exito

false Error

Definition at line 316 of file fsm.c.

4.3.5 Variable Documentation

4.3.5.1 en_game_over STATE en_game_over

Initial value:

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

Definition at line 295 of file fsm.c.

4.3.5.2 en_game_over_esperando_opcion STATE en_game_over_esperando_opcion

Initial value:

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

Definition at line 303 of file fsm.c.

4.3.5.3 en_menu_ppal STATE en_menu_ppal

Initial value:

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

Definition at line 219 of file fsm.c.

4.3.5.4 en_pausa STATE en_pausa

Initial value:

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

Definition at line 280 of file fsm.c.

4.3.5.5 en_pausa_esperando_opcion STATE en_pausa_esperando_opcion

Initial value:

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

Definition at line 288 of file fsm.c.

4.3.5.6 jugando STATE jugando

Initial value:

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

Definition at line 269 of file fsm.c.

4.3.5.7 menu_ppal_esperando_opcion STATE menu_ppal_esperando_opcion

Initial value:

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

Definition at line 227 of file fsm.c.

4.3.5.8 poniendo_nombre STATE poniendo_nombre

Initial value:

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

Definition at line 258 of file fsm.c.

4.3.5.9 seleccionando_dificultad STATE seleccionando_dificultad

Initial value:

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

Definition at line 238 of file fsm.c.

4.3.5.10 viendo_creditos STATE viendo_creditos

Initial value:

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

Definition at line 252 of file fsm.c.

4.3.5.11 viendo_ranking STATE viendo_ranking

Initial value:

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

Definition at line 246 of file fsm.c.

4.4 fsm.c

Go to the documentation of this file.

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

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

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

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

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

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

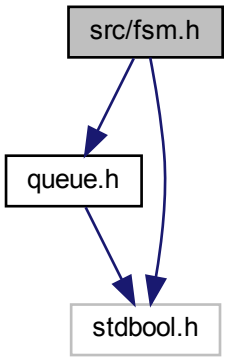
```
00556     int menu[3] = {CONTINUAR, REINICIAR, SALIRTXT};
00557     setMenu(menu, 3);
00558     setOpcion(0);
00559 }
00560
00561 static void continuar(void)
00562 {
00563     limpiarDisplay();
00564     reconfigurarDisplayOFF();
00565     reproducirMusica(MUSICA_JUGANDO);
00566     reanudarJuego();
00567     pthread_create(&tjuego, NULL, threadJuego, NULL);
00568 }
00569
00570 static void procesar_game_over(void)
00571 {
00572     pthread_join(tjuego, NULL);
00573
00574     limpiarDisplay();
00575
00576     reproducirMusica(MUSICA_GAME_OVER);
00577     reconfigurarDisplayON();
00578
00579     unsigned long long jugador_puntos = getPuntos();
00580
00581     if (jugador_puntos > getMaxPuntos())
00582     {
00583         reproducirEfecto(EFECTO_NUEVO_MAX_SCORE);
00584
00585         mostrarTexto("NUEVA PUNTUACION ALTA", POS_MSJ_NEW_HI_SCORE);
00586         setMaxPuntos(jugador_puntos);
00587
00588         actualizarRanking(getNombre(), getMaxPuntos());
00589     }
00590
00591     mostrarTexto("FIN DEL JUEGO", POS_MSJ_GAME_OVER);
00592     int menu[2] = {REINICIAR, SALIRTXT};
00593     limpiarDisplay();
00594     setMenu(menu, 2);
00595     setOpcion(0);
00596 }
```

4.5 src/fsm.h File Reference

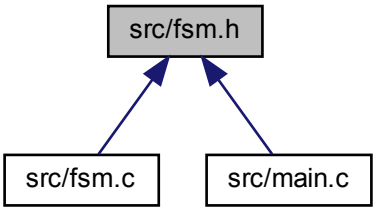
Header del modulo fsm. Contiene los prototipos de funciones necesarias para iniciar la fsm desde main.c.

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

Include dependency graph for fsm.h:



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



Functions

- bool inicializarFsm (void)

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

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

  Fixea consumo elevado de cpu en el while loop principal.

4.5.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file fsm.h.

4.5.2 Function Documentation

4.5.2.1 fixHighCpuUsage() `void fixHighCpuUsage ( void )`

Fixea consumo elevado de cpu en el while loop principal.

Not the best solucion, but sirve...

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

Definition at line 356 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 338 of file fsm.c.

4.5.2.3 inicializarFsm() `bool inicializarFsm ( void )`

Inicializa la FSM, notificando si tuvo exito.

Returns

true Exito

false Error

Definition at line 316 of file fsm.c.

4.6 fsm.h

Go to the documentation of this file.

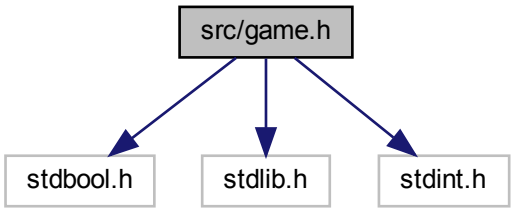
```
00001
00013 #ifndef _FSM_H_
00014 #define _FSM_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include "queue.h"
00021 #include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

4.7 src/game.h File Reference

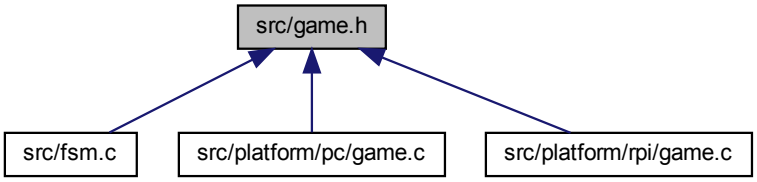
Header del modulo game Vinculo entre la fsm y las plataformas en lo que respecta a la informacion del jugador y el progreso del juego.

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

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



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



Functions

- void setNombre (char *nombre)
  Confirma el nombre del jugador.
- void setMaxPuntos (unsigned long long max)
  Setea los puntos maximos del jugador.
- void setDificultad (int dif)
  Setea la dificultad a usar.
- bool tiempoRefrescoEntidades (void)
  Chequea si es tiempo de refrescar entidades según la plataforma.
- char * getNombre (void)
  Devuelve el nombre del jugador.
- unsigned long long getPuntos (void)
  Devuelve el puntaje del jugador.
- unsigned long long getMaxPuntos (void)
  Devuelve el puntaje máximo del jugador.
- int getNivel (void)
  Devuelve el nivel//run del jugador.
- void inicializarJuego (void)
  Rutina que se ejecuta al comenzar a jugar, luego de ingresar el nombre.

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

4.7.1 Detailed Description

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

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

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 188 of file game.c.

4.7.2.14 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 114 of file game.c.

4.7.2.15 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 105 of file game.c.

4.7.2.16 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 151 of file game.c.

4.7.2.17 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 47 of file game.c.

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

Setea los puntos maximos del jugador.

Parameters

max	

Definition at line 42 of file game.c.

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

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 37 of file game.c.

4.7.2.20 tiempoRefrescoEntidades() `bool tiempoRefrescoEntidades ( void )`

Chequea si es tiempo de refrescar entidades según la plataforma.

Returns

true

false

Definition at line 67 of file game.c.

4.8 game.h

Go to the documentation of this file.

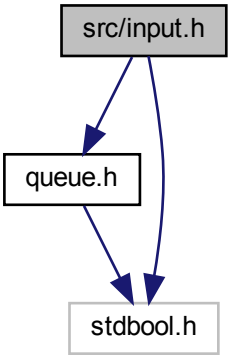
```
00001
00013 #ifndef _GAME_H_
00014 #define _GAME_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdbool.h>
00021 #include <stdlib.h>
00022 #include <stdint.h>
00023
00024
00025 /*****************************************************************************
00026  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00027  *****************************************************************************/
00028
00034 void setNombre(char* nombre);
00035
00036
00042 void setMaxPuntos(unsigned long long max);
00043
00049 void setDificultad(int dif);
00050
00057 bool tiempoRefrescoEntidades(void);
00058
00064 char* getNombre(void);
00065
00071 unsigned long long getPuntos(void);
00072
00078 unsigned long long getMaxPuntos(void);
00079
00085 int getNivel(void);
00086
00091 void inicializarJuego(void);
00092
00097 void reiniciarNivel(void);
00098
00103 void pausarJuego(void);
00104
00109 void reanudarJuego(void);
00110
00115 void refrescar(void);
00116
00121 void moverAdelante(void);
00122
00127 void moverAtras(void);
00128
00133 void moverIzda(void);
00134
00139 void moverDcha(void);
00140
00145 void respawn(void);
00146
00151 void perderVida(void);
00152
00157 void actualizarInterfaz(void);
00158
00159
00160 /*****************************************************************************
00161  *****************************************************************************/
00162
00163 #endif // _GAME_H_
```

4.9 src/input.h File Reference

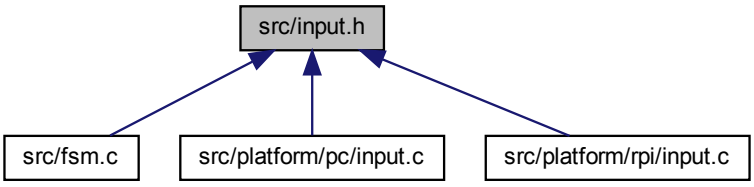
Header del modulo input Vinculo entre la fsm y las plataformas en lo que respecta al manejo de acciones externas.

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

Include dependency graph for input.h:



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



Functions

- void iniciarEntradas (void)

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

  Devuelve una entrada válida.

4.9.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file input.h.

4.9.2 Function Documentation

4.9.2.1 iniciarEntradas() `void iniciarEntradas ( void )`

Inicializa las entradas de la plataforma.

Definition at line 36 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 40 of file input.c.

4.10 input.h

Go to the documentation of this file.

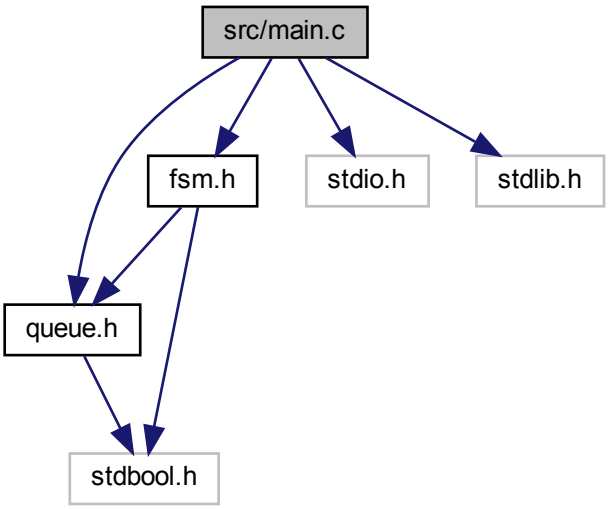
```
00001
00013 #ifndef _INPUT_H_
00014 #define _INPUT_H_
00015
00016 /********************************************************************************
00017  * INCLUDE HEADER FILES
00018  *******************************************************************************/
00019
00020 #include "queue.h"
00021
00022 #include <stdbool.h>
00023
00024
00025 /********************************************************************************
00026  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00027  *******************************************************************************/
00028
00034 void iniciarEntradas(void);
00035
00041 event_t leerEntradas(void);
00042
00043
00044 /********************************************************************************
00045  *******************************************************************************/
00046
00047 #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 34 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  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00025  *****************************************************************************/
00026
00027
00028 /*****************************************************************************
00029  *****************************************************************************
00030                                     MAIN
00031  *****************************************************************************
00032  *****************************************************************************/
00033
00034 int main (void)
00035 {
00036     event_t evento;
00037
00038     if(!inicializarFsm())
00039         return 1;
00040
00041     while((evento = queueSiguienteEvento()))
00042     {
00043         if(evento != NADA)
00044         {
00045             fsm(evento);
00046         }
00047
00048         fixHighCpuUsage();
00049     }
00050
00051     destruirQueue();
00052
00053     printf("\nSaliendo...\n");
00054
00055     return 0;
00056 }
00057
00058
00059 /*****************************************************************************
00060  *****************************************************************************
00061                          LOCAL FUNCTION DEFINITIONS
00062  *****************************************************************************
00063  *****************************************************************************/
00064
```

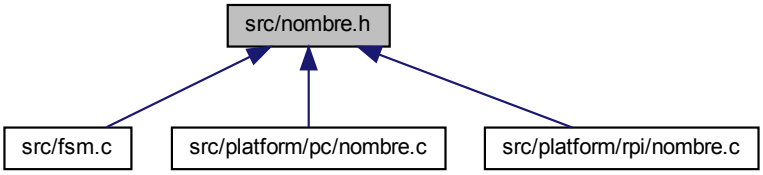
4.13 menu.h

```
00001
00012 #ifndef _MENU_H_
00013 #define _MENU_H_
00014
00015 /*******************************************************************************
00016  * INCLUDE HEADER FILES
00017  ******************************************************************************/
00018
00019 #include "queue.h"
00020
00021
00022 /*******************************************************************************
00023  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00024  ******************************************************************************/
00025
00026 enum textos_menu
00027 {
00028     JUGAR = 0,
00029     DIFICULTAD,
00030     RANKING,
00031     CREDITOS,
00032     SALIRTXT,
00033     CONTINUAR,
00034     REINICIAR,
00035     FACIL,
00036     NORMAL,
00037     DIFICIL
00038 };
00039
00040
00041 /*******************************************************************************
00042  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00043  ******************************************************************************/
00044
00049 void iniciarMenu(void);
00050
00055 void destruirMenu(void);
00056
00063 void setMenu(int* a, unsigned int size);
00064
00070 void setOpcion(int opc);
00071
00077 int getOpcion(void);
00078
00083 void subirOpcion(void);
00084
00089 void bajarOpcion(void);
00090
00091 /*******************************************************************************
00092  ******************************************************************************/
00093
00094 #endif // _MENU_H_
```

4.14 src/nombre.h File Reference

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

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



Functions

- void nuevoNombre (void)

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

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

  Selecciona la letra inferior.
- void siguienteLetra (void)

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

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

  Devuelve puntero al nombre.

4.14.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file nombre.h.

4.14.2 Function Documentation

4.14.2.1 agregarLetra() `void agregarLetra ( void )`

Confirma la letra.

Definition at line 61 of file nombre.c.

4.14.2.2 bajarLetra() `void bajarLetra ( void )`

Selecciona la letra inferior.

Definition at line 53 of file nombre.c.

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

Devuelve puntero al nombre.

Returns

char* Puntero al nombre

Definition at line 82 of file nombre.c.

4.14.2.4 nuevoNombre() `void nuevoNombre ( void )`

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

Definition at line 36 of file nombre.c.

4.14.2.5 siguienteLetra() `void siguienteLetra ( void )`

Confirma la letra y pasa a seleccionar la siguiente.

Definition at line 57 of file nombre.c.

4.14.2.6 subirLetra() `void subirLetra ( void )`

Selecciona la siguiente letra superior.

Definition at line 49 of file nombre.c.

4.14.2.7 subirNombre() `void subirNombre ( void )`

Definition at line 78 of file nombre.c.

4.15 nombre.h

Go to the documentation of this file.

```
00001
00013 #ifndef _NOMBRE_H_
00014 #define _NOMBRE_H_
00015
00016
00017 /*****************************************************************************
00018  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00019  *****************************************************************************/
00020
00025 void nuevoNombre(void);
00026
00031 void subirLetra(void);
00032
00037 void bajarLetra(void);
00038
00043 void siguienteLetra(void);
00044
00049 void agregarLetra(void);
00050
00055 void subirNombre(void);
00056
00062 char* devolverNombre(void);
00063
00064
00065 /*****************************************************************************
00066  *****************************************************************************/
00067
00068 #endif // _NOMBRE_H_
```

4.16 algif.c

```
00001 #include "algif.h"
00002 #include <allegro5/allegro_primitives.h>
00003 #include <math.h>
00004
00005 /* Renders the next frame in a GIF animation to the given position.
00006  * You need to call this in order on the same destination for frames
00007  * [0..gif->frames_count - 1] to properly render all the frames in the GIF.
00008  * The current target bitmap should have the same height as the animation,
00009  * and blending should be set to fully copy RGBA.
00010  */
00011 void algif_render_frame(ALGIF_ANIMATION *gif, int frame, int xpos, int ypos)
00012 {
00013   int x, y, w, h;
00014   ALGIF_FRAME *f = &gif->frames[frame];
00015   ALGIF_PALETTE *pal;
00016   if (frame == 0)
00017   {
00018     al_draw_filled_rectangle(xpos, ypos, xpos + gif->width,
00019                              ypos + gif->height, al_map_rgba_f(0, 0, 0, 0));
00020   }
00021   else
00022   {
00023     ALGIF_FRAME *p = &gif->frames[frame - 1];
00024     if (p->disposal_method == 2)
00025     {
00026       al_draw_filled_rectangle(xpos + p->xoff, ypos + p->yoff,
00027                                xpos + p->xoff + p->bitmap_8_bit->w,
00028                                ypos + p->yoff + p->bitmap_8_bit->h,
00029                                al_map_rgba_f(0, 0, 0, 0));
00030     }
00031     else if (p->disposal_method == 3 && gif->store)
00032     {
00033       al_draw_bitmap_region(gif->store, xpos + p->xoff, ypos + p->yoff,
00034                             p->bitmap_8_bit->w,
00035                             p->bitmap_8_bit->h,
00036                             xpos + p->xoff, ypos + p->yoff, 0);
00037       al_destroy_bitmap(gif->store);
00038       gif->store = NULL;
00039     }
00040   }
00041   w = f->bitmap_8_bit->w;
00042   h = f->bitmap_8_bit->h;
00043   if (f->disposal_method == 3)
00044   {
00045     if (gif->store)
00046       al_destroy_bitmap(gif->store);
00047     gif->store = al_clone_bitmap(al_get_target_bitmap());
00048   }
```

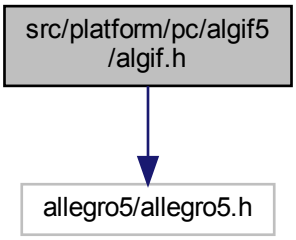
```
00049   pal = &gif->frames[frame].palette;
00050   if (pal->colors_count == 0)
00051     pal = &gif->palette;
00052
00053   for (y = 0; y < h; y++)
00054   {
00055     for (x = 0; x < w; x++)
00056     {
00057       int c = f->bitmap_8_bit->data[x + y * f->bitmap_8_bit->w];
00058       if (c != f->transparent_index)
00059       {
00060         al_draw_pixel(xpos + f->xoff + x, ypos + f->yoff + y,
00061                       al_map_rgb(pal->colors[c].r, pal->colors[c].g,
00062                                  pal->colors[c].b));
00063       }
00064     }
00065   }
00066 }
00067
00068 ALGIF_ANIMATION *algif_load_animation_f(ALLEGRO_FILE *file)
00069 {
00070   ALGIF_ANIMATION *gif = algif_load_raw(file);
00071
00072   if (!gif)
00073     return gif;
00074
00075   al_init_primitives_addon();
00076
00077   gif->duration = 0;
00078   ALLEGRO_STATE s;
00079   al_store_state(&s, ALLEGRO_STATE_TARGET_BITMAP | ALLEGRO_STATE_BLENDER);
00080   al_set_blender(ALLEGRO_ADD, ALLEGRO_ONE, ALLEGRO_ZERO);
00081   int n = gif->frames_count;
00082   int i;
00083   for (i = 0; i < n; i++)
00084   {
00085     ALGIF_FRAME *f = &gif->frames[i];
00086     if (i == 0)
00087       f->rendered = al_create_bitmap(gif->width, gif->height);
00088     else
00089       f->rendered = al_clone_bitmap(gif->frames[i - 1].rendered);
00090     al_set_target_bitmap(f->rendered);
00091     algif_render_frame(gif, i, 0, 0);
00092     gif->duration += f->duration;
00093   }
00094
00095   al_restore_state(&s);
00096   return gif;
00097 }
00098
00099 ALGIF_ANIMATION *algif_load_animation(char const *filename)
00100 {
00101   ALLEGRO_FILE *file = al_fopen(filename, "rb");
00102   return algif_load_animation_f(file);
00103 }
00104
00105 ALLEGRO_BITMAP *algif_get_bitmap(ALGIF_ANIMATION *gif, double seconds)
00106 {
00107   int n = gif->frames_count;
00108   seconds = fmod(seconds, gif->duration / 100.0);
00109   double d = 0;
00110   int i;
00111   for (i = 0; i < n; i++)
00112   {
00113     d += gif->frames[i].duration / 100.0;
00114     if (seconds < d)
00115       return gif->frames[i].rendered;
00116   }
00117   return gif->frames[0].rendered;
00118 }
00119
00120 ALLEGRO_BITMAP *algif_get_frame_bitmap(ALGIF_ANIMATION *gif, int i)
00121 {
00122   return gif->frames[i].rendered;
00123 }
00124
00125 double algif_get_frame_duration(ALGIF_ANIMATION *gif, int i)
00126 {
00127   return gif->frames[i].duration / 100.0;
00128 }
```

4.17 src/platform/pc/algif5/algif.h File Reference

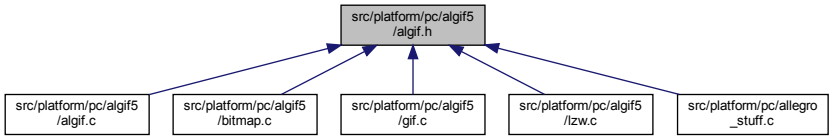
Header para el uso de la libreria algif (algif5 is a gif loading library for Allegro 5)

```
#include <allegro5/allegro5.h>
```

Include dependency graph for algif.h:



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



Data Structures

- struct ALGIF_RGB
- struct ALGIF_PALETTE
- struct ALGIF_BITMAP
- struct ALGIF_ANIMATION
- struct ALGIF_FRAME

Typedefs

- typedef struct ALGIF_ANIMATION ALGIF_ANIMATION
- typedef struct ALGIF_FRAME ALGIF_FRAME
- typedef struct ALGIF_PALETTE ALGIF_PALETTE
- typedef struct ALGIF_BITMAP ALGIF_BITMAP
- typedef struct ALGIF_RGB ALGIF_RGB

Functions

- ALGIF_ANIMATION * algif_load_raw (ALLEGRO_FILE *file)
- ALGIF_ANIMATION * algif_load_animation_f (ALLEGRO_FILE *file)
- ALGIF_ANIMATION * algif_load_animation (char const *filename)
- void algif_render_frame (ALGIF_ANIMATION *gif, int frame, int xpos, int ypos)
- void algif_destroy_animation (ALGIF_ANIMATION *gif)

- ALGIF_BITMAP * algif_create_bitmap (int w, int h)
- void algif_destroy_bitmap (ALGIF_BITMAP *bitmap)
- void algif_blit (ALGIF_BITMAP *from, ALGIF_BITMAP *to, int xf, int yf, int xt, int yt, int w, int h)
- ALLEGRO_BITMAP * algif_get_bitmap (ALGIF_ANIMATION *gif, double seconds)
- ALLEGRO_BITMAP * algif_get_frame_bitmap (ALGIF_ANIMATION *gif, int i)
- double algif_get_frame_duration (ALGIF_ANIMATION *gif, int i)

4.17.1 Detailed Description

Header para el uso de la libreria algif (algif5 is a gif loading library for Allegro 5)

Author

allefant (https://github.com/allefant)

https://github.com/allefant/algif5

Definition in file algif.h.

4.17.2 Typedef Documentation

4.17.2.1 ALGIF_ANIMATION `typedef struct ALGIF_ANIMATION ALGIF_ANIMATION`

Definition at line 20 of file algif.h.

4.17.2.2 ALGIF_BITMAP `typedef struct ALGIF_BITMAP ALGIF_BITMAP`

Definition at line 23 of file algif.h.

4.17.2.3 ALGIF_FRAME `typedef struct ALGIF_FRAME ALGIF_FRAME`

Definition at line 21 of file algif.h.

4.17.2.4 ALGIF_PALETTE `typedef struct ALGIF_PALETTE ALGIF_PALETTE`

Definition at line 22 of file algif.h.

4.17.2.5 ALGIF_RGB `typedef struct ALGIF_RGB ALGIF_RGB`

Definition at line 24 of file algif.h.

4.17.3 Function Documentation

4.17.3.1 algif_blit() `void algif_blit (`

```
ALGIF_BITMAP * from,
ALGIF_BITMAP * to,
int xf,
int yf,
int xt,
int yt,
int w,
int h )
```

Definition at line 18 of file bitmap.c.

4.17.3.2 algif_create_bitmap() `ALGIF_BITMAP * algif_create_bitmap (`

```
int w,
int h )
```

Definition at line 3 of file bitmap.c.

4.17.3.3 algif_destroy_animation() `void algif_destroy_animation (`

```
ALGIF_ANIMATION * gif )
```

Definition at line 9 of file gif.c.

4.17.3.4 algif_destroy_bitmap() `void algif_destroy_bitmap (`

```
ALGIF_BITMAP * bitmap )
```

Definition at line 12 of file bitmap.c.

4.17.3.5 algif_get_bitmap() `ALLEGRO_BITMAP * algif_get_bitmap (`

```
ALGIF_ANIMATION * gif,
double seconds )
```

Definition at line 105 of file algif.c.

4.17.3.6 algif_get_frame_bitmap()

```
ALLEGRO_BITMAP * algif_get_frame_bitmap (
    ALGIF_ANIMATION * gif,
    int i )
```

Definition at line 120 of file algif.c.

4.17.3.7 algif_get_frame_duration()

```
double algif_get_frame_duration (
    ALGIF_ANIMATION * gif,
    int i )
```

Definition at line 125 of file algif.c.

4.17.3.8 algif_load_animation()

```
ALGIF_ANIMATION * algif_load_animation (
    char const * filename )
```

Definition at line 99 of file algif.c.

4.17.3.9 algif_load_animation_f()

```
ALGIF_ANIMATION * algif_load_animation_f (
    ALLEGRO_FILE * file )
```

Definition at line 68 of file algif.c.

4.17.3.10 algif_load_raw()

```
ALGIF_ANIMATION * algif_load_raw (
    ALLEGRO_FILE * file )
```

Definition at line 64 of file gif.c.

4.17.3.11 algif_render_frame()

```
void algif_render_frame (
    ALGIF_ANIMATION * gif,
    int frame,
    int xpos,
    int ypos )
```

Definition at line 11 of file algif.c.

4.18 algif.h

Go to the documentation of this file.

```
00001
00010 #ifndef _GIF_H_
00011 #define _GIF_H_
00012
00013 #include <allegro5/allegro5.h>
00014
00015 #ifdef __cplusplus
00016 extern "C"
00017 {
00018 #endif
00019
00020   typedef struct ALGIF_ANIMATION ALGIF_ANIMATION;
00021   typedef struct ALGIF_FRAME ALGIF_FRAME;
00022   typedef struct ALGIF_PALETTE ALGIF_PALETTE;
00023   typedef struct ALGIF_BITMAP ALGIF_BITMAP;
00024   typedef struct ALGIF_RGB ALGIF_RGB;
00025
00026   struct ALGIF_RGB
00027   {
00028     uint8_t r, g, b;
00029   };
00030
00031   struct ALGIF_PALETTE
00032   {
00033     int colors_count;
00034     ALGIF_RGB colors[256];
00035   };
00036
00037   struct ALGIF_BITMAP
00038   {
00039     int w, h;
00040     uint8_t *data;
00041   };
00042
00043   struct ALGIF_ANIMATION
00044   {
00045     int width, height;
00046     int frames_count;
00047     int background_index;
00048     int loop; /* -1 = no, 0 = forever, 1..65535 = that many times */
00049     ALGIF_PALETTE palette;
00050     ALGIF_FRAME *frames;
00051
00052     int duration;
00053     ALLEGRO_BITMAP *store;
00054   };
00055
00056   struct ALGIF_FRAME
00057   {
00058     ALGIF_BITMAP *bitmap_8_bit;
00059     ALGIF_PALETTE palette;
00060     int xoff, yoff;
00061     int duration;        /* in 1/100th seconds */
00062     int disposal_method; /* 0 = don't care, 1 = keep, 2 = background, 3 = previous */
00063     int transparent_index;
00064
00065     ALLEGRO_BITMAP *rendered;
00066   };
00067
00068   ALGIF_ANIMATION *algif_load_raw(ALLEGRO_FILE *file);
00069   ALGIF_ANIMATION *algif_load_animation_f(ALLEGRO_FILE *file);
00070   ALGIF_ANIMATION *algif_load_animation(char const *filename);
00071   void algif_render_frame(ALGIF_ANIMATION *gif, int frame, int xpos, int ypos);
00072   void algif_destroy_animation(ALGIF_ANIMATION *gif);
00073
00074   ALGIF_BITMAP *algif_create_bitmap(int w, int h);
00075   void algif_destroy_bitmap(ALGIF_BITMAP *bitmap);
00076   void algif_blit(ALGIF_BITMAP *from, ALGIF_BITMAP *to, int xf, int yf, int xt, int yt,
00077                   int w, int h);
00078   ALLEGRO_BITMAP *algif_get_bitmap(ALGIF_ANIMATION *gif, double seconds);
00079   ALLEGRO_BITMAP *algif_get_frame_bitmap(ALGIF_ANIMATION *gif, int i);
00080   double algif_get_frame_duration(ALGIF_ANIMATION *gif, int i);
00081
00082 #ifdef __cplusplus
00083 }
00084 #endif
00085
00086 #endif //_GIF_H_
```

4.19 bitmap.c

```
00001 #include "algif.h"
00002
00003 ALGIF_BITMAP *algif_create_bitmap(int w, int h)
00004 {
00005   ALGIF_BITMAP *bitmap = calloc(1, sizeof *bitmap);
00006   bitmap->w = w;
00007   bitmap->h = h;
00008   bitmap->data = calloc(1, w * h);
00009   return bitmap;
00010 }
00011
00012 void algif_destroy_bitmap(ALGIF_BITMAP *bitmap)
00013 {
00014   free(bitmap->data);
00015   free(bitmap);
00016 }
00017
00018 void algif_blit(ALGIF_BITMAP *from, ALGIF_BITMAP *to, int xf, int yf, int xt, int yt,
00019                 int w, int h)
00020 {
00021
00022   if (w <= 0 || h <= 0)
00023     return;
00024
00025   /* source clipping */
00026   if (xf < 0)
00027   {
00028     w += xf;
00029     xt -= xf;
00030     xf = 0;
00031   }
00032   if (yf < 0)
00033   {
00034     h += yf;
00035     yt -= yf;
00036     yf = 0;
00037   }
00038   int wf = from->w;
00039   int hf = from->h;
00040   if (xf + w > wf)
00041   {
00042     w = wf - xf;
00043   }
00044   if (yf + h > hf)
00045   {
00046     h = hf - yf;
00047   }
00048
00049   /* destination clipping */
00050   if (xt < 0)
00051   {
00052     w += xt;
00053     xf -= xt;
00054     xt = 0;
00055   }
00056   if (yt < 0)
00057   {
00058     h += yt;
00059     yf -= yt;
00060     yt = 0;
00061   }
00062   int wt = to->w;
00063   int ht = to->h;
00064   if (xt + w > wt)
00065   {
00066     w = wt - xt;
00067   }
00068   if (yt + h > ht)
00069   {
00070     h = ht - yt;
00071   }
00072
00073   if (w <= 0 || h <= 0)
00074     return;
00075
00076   /* copy row by row */
00077   uint8_t *pf = from->data + yf * from->w;
00078   uint8_t *pt = to->data + yt * to->w;
00079   int i;
00080   for (i = 0; i < h; i++)
00081   {
00082     memmove(pt + xt, pf + xf, w);
00083     pf += from->w;
00084     pt += to->w;
00085   }
```

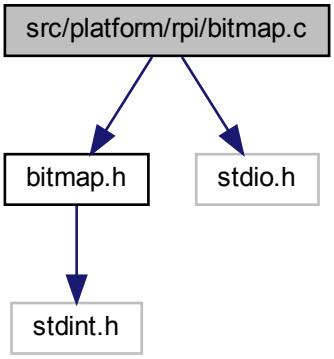
```
00086 }
```

4.20 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".

4.20.1 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.

4.20.2 Function Documentation

4.20.2.1 copiarMatriz()

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

Copia el contenido de una matriz en otra.

Parameters

destino	
desde	

Definition at line 37 of file bitmap.c.

4.20.2.2 limpiarMatriz()

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

Borra el contenido de una matriz.

Parameters

A	

Definition at line 31 of file bitmap.c.

4.20.2.3 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 43 of file bitmap.c.

4.20.2.4 matrizNot()

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

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

Parameters

A	

Definition at line 61 of file bitmap.c.

4.20.2.5 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 49 of file bitmap.c.

4.20.2.6 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 55 of file bitmap.c.

4.20.2.7 printMatriz() void printMatriz (matriz_t A)

Imprime una matriz en consola (para debug)

Parameters

A	

Definition at line 22 of file bitmap.c.

4.21 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   for (int i = 0; i < CANT_FILAS; i++, putchar('\n'))
00025     for (int j = 0; j < CANT_FILAS; j++)
00026       putchar((a[i] & (0b1000000000000000 >> j)) ? '1' : '.');
00027
00028   putchar('\n');
00029 }
00030
00031 void limpiarMatriz(matriz_t a)
00032 {
00033   for (int i = 0; i < CANT_FILAS; i++)
00034     a[i] = 0;
00035 }
00036
00037 void copiarMatriz(matriz_t destino, const matriz_t desde)
00038 {
00039   for (int i = 0; i < CANT_FILAS; i++)
00040     destino[i] = desde[i];
00041 }
00042
00043 void matrizAnd(matriz_t a, matriz_t b)
00044 {
00045   for (int i = 0; i < CANT_FILAS; i++)
00046     a[i] &= b[i];
00047 }
00048
00049 void matrizOr(matriz_t a, matriz_t b)
00050 {
```

```
00051   for (int i = 0; i < CANT_FILAS; i++)
00052     a[i] |= b[i];
00053 }
00054
00055 void matrizXor(matriz_t a, matriz_t b)
00056 {
00057   for (int i = 0; i < CANT_FILAS; i++)
00058     a[i] ^= b[i];
00059 }
00060
00061 void matrizNot(matriz_t a)
00062 {
00063   for (int i = 0; i < CANT_FILAS; i++)
00064     a[i] = ~a[i];
00065 }
```

4.22 gif.c

```
00001 #include "algif.h"
00002
00003 #include <stdlib.h>
00004 #include <string.h>
00005
00006 int LZW_decode(ALLEGRO_FILE *file, ALGIF_BITMAP *bmp);
00007
00008 /* Destroy a complete gif, including all frames. */
00009 void algif_destroy_animation(ALGIF_ANIMATION *gif)
00010 {
00011   int i;
00012
00013   for (i = 0; i < gif->frames_count; i++)
00014   {
00015     ALGIF_FRAME *frame = gif->frames + i;
00016
00017     if (frame->bitmap_8_bit)
00018       algif_destroy_bitmap(frame->bitmap_8_bit);
00019     if (frame->rendered)
00020       al_destroy_bitmap(frame->rendered);
00021   }
00022   if (gif->store)
00023     al_destroy_bitmap(gif->store);
00024   free(gif->frames);
00025   free(gif);
00026 }
00027
00028 static void read_palette(ALLEGRO_FILE *file, ALGIF_PALETTE *palette)
00029 {
00030   int i;
00031
00032   for (i = 0; i < palette->colors_count; i++)
00033   {
00034     palette->colors[i].r = al_fgetc(file);
00035     palette->colors[i].g = al_fgetc(file);
00036     palette->colors[i].b = al_fgetc(file);
00037   }
00038 }
00039
00040 static void deinterlace(ALGIF_BITMAP *bmp)
00041 {
00042   ALGIF_BITMAP *n = algif_create_bitmap(bmp->w, bmp->h);
00043   int y, i = 0;
00044   for (y = 0; y < n->h; y += 8)
00045   {
00046     algif_blit(bmp, n, 0, i++, 0, y, n->w, 1);
00047   }
00048   for (y = 4; y < n->h; y += 8)
00049   {
00050     algif_blit(bmp, n, 0, i++, 0, y, n->w, 1);
00051   }
00052   for (y = 2; y < n->h; y += 4)
00053   {
00054     algif_blit(bmp, n, 0, i++, 0, y, n->w, 1);
00055   }
00056   for (y = 1; y < n->h; y += 2)
00057   {
00058     algif_blit(bmp, n, 0, i++, 0, y, n->w, 1);
00059   }
00060   algif_blit(n, bmp, 0, 0, 0, 0, n->w, n->h);
00061   algif_destroy_bitmap(n);
00062 }
00063
00064 ALGIF_ANIMATION *algif_load_raw(ALLEGRO_FILE *file)
00065 {
00066   if (!file)
```

```
00067     return NULL;
00068
00069   int version;
00070   ALGIF_BITMAP *bmp = NULL;
00071   int i, j;
00072   ALGIF_ANIMATION *gif = calloc(1, sizeof *gif);
00073   ALGIF_FRAME frame;
00074
00075   gif->frames_count = 0;
00076
00077   /* is it really a GIF? */
00078   if (al_fgetc(file) != 'G')
00079     goto error;
00080   if (al_fgetc(file) != 'I')
00081     goto error;
00082   if (al_fgetc(file) != 'F')
00083     goto error;
00084   if (al_fgetc(file) != '8')
00085     goto error;
00086   /* '7' or '9', for 87a or 89a. */
00087   version = al_fgetc(file);
00088   if (version != '7' && version != '9')
00089     goto error;
00090   if (al_fgetc(file) != 'a')
00091     goto error;
00092
00093   gif->width = al_fread16le(file);
00094   gif->height = al_fread16le(file);
00095   i = al_fgetc(file);
00096   /* Global color table? */
00097   if (i & 128)
00098     gif->palette.colors_count = 1 << ((i & 7) + 1);
00099   else
00100     gif->palette.colors_count = 0;
00101   /* Background color is only valid with a global palette. */
00102   gif->background_index = al_fgetc(file);
00103
00104   /* Skip aspect ratio. */
00105   al_fseek(file, 1, ALLEGRO_SEEK_CUR);
00106
00107   if (gif->palette.colors_count)
00108   {
00109     read_palette(file, &gif->palette);
00110   }
00111
00112   memset(&frame, 0, sizeof frame); /* For first frame. */
00113   frame.transparent_index = -1;
00114
00115   do
00116   {
00117     i = al_fgetc(file);
00118
00119     switch (i)
00120     {
00121     case 0x2c: /* Image Descriptor */
00122     {
00123       int w, h;
00124       int interlaced = 0;
00125
00126       frame.xoff = al_fread16le(file);
00127       frame.yoff = al_fread16le(file);
00128       w = al_fread16le(file);
00129       h = al_fread16le(file);
00130       bmp = algif_create_bitmap(w, h);
00131       if (!bmp)
00132         goto error;
00133       i = al_fgetc(file);
00134
00135       /* Local palette. */
00136       if (i & 128)
00137       {
00138         frame.palette.colors_count = 1 << ((i & 7) + 1);
00139         read_palette(file, &frame.palette);
00140       }
00141       else
00142       {
00143         frame.palette.colors_count = 0;
00144       }
00145
00146       if (i & 64)
00147         interlaced = 1;
00148
00149       if (LZW_decode(file, bmp))
00150         goto error;
00151
00152       if (interlaced)
00153         deinterlace(bmp);
```

```
00154
00155       frame.bitmap_8_bit = bmp;
00156       bmp = NULL;
00157
00158       gif->frames_count++;
00159       gif->frames =
00160           realloc(gif->frames,
00161                   gif->frames_count * sizeof *gif->frames);
00162       gif->frames[gif->frames_count - 1] = frame;
00163
00164       memset(&frame, 0, sizeof frame); /* For next frame. */
00165       frame.transparent_index = -1;
00166
00167       break;
00168     }
00169     case 0x21:            /* Extension Introducer. */
00170       j = al_fgetc(file); /* Extension Type. */
00171       i = al_fgetc(file); /* Size. */
00172       if (j == 0xf9)      /* Graphic Control Extension. */
00173       {
00174         /* size must be 4 */
00175         if (i != 4)
00176           goto error;
00177         i = al_fgetc(file);
00178         frame.disposal_method = (i >> 2) & 7;
00179         frame.duration = al_fread16le(file);
00180         if (i & 1) /* Transparency? */
00181         {
00182           frame.transparent_index = al_fgetc(file);
00183         }
00184         else
00185         {
00186           al_fseek(file, 1, ALLEGRO_SEEK_CUR);
00187           frame.transparent_index = -1;
00188         }
00189         i = al_fgetc(file); /* Size. */
00190       }
00191       /* Application Extension. */
00192       else if (j == 0xff)
00193       {
00194         if (i == 11)
00195         {
00196           char name[12];
00197           al_fread(file, name, 11);
00198           i = al_fgetc(file); /* Size. */
00199           name[11] = '\0';
00200           if (!strcmp(name, "NETSCAPE2.0"))
00201           {
00202             if (i == 3)
00203             {
00204               j = al_fgetc(file);
00205               gif->loop = al_fread16le(file);
00206               if (j != 1)
00207                 gif->loop = 0;
00208               i = al_fgetc(file); /* Size. */
00209             }
00210           }
00211         }
00212       }
00213
00214       /* Possibly more blocks until terminator block (0). */
00215       while (i)
00216       {
00217         al_fseek(file, i, ALLEGRO_SEEK_CUR);
00218         i = al_fgetc(file);
00219       }
00220       break;
00221     case 0x3b:
00222       /* GIF Trailer. */
00223       al_fclose(file);
00224       return gif;
00225     }
00226   } while (true);
00227 error:
00228   if (file)
00229     al_fclose(file);
00230   if (gif)
00231     algif_destroy_animation(gif);
00232   if (bmp)
00233     algif_destroy_bitmap(bmp);
00234   return NULL;
00235 }
```

4.23 lzw.c

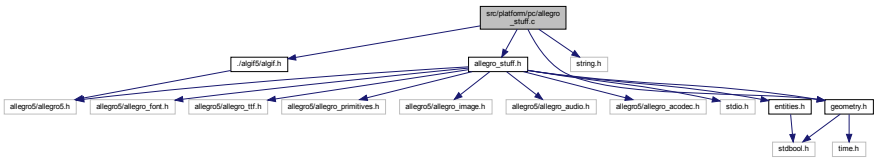
```
00001 #include "algif.h"
00002
00003 static int
00004 read_code(ALLEGRO_FILE *file, char *buf, int *bit_pos, int bit_size)
00005 {
00006   int i;
00007   int code = 0;
00008   int pos = 1;
00009
00010   for (i = 0; i < bit_size; i++)
00011   {
00012     int byte_pos = (*bit_pos » 3) & 255;
00013
00014     if (byte_pos == 0)
00015     {
00016       int data_len = al_fgetc(file);
00017
00018       if (data_len == 0)
00019       {
00020         // printf ("Fatal. Errorneous GIF stream.\n");
00021         // abort ();
00022         return -1;
00023       }
00024       al_fread(file, buf + 256 - data_len, data_len);
00025       byte_pos = 256 - data_len;
00026       *bit_pos = byte_pos « 3;
00027     }
00028     if (buf[byte_pos] & (1 « (*bit_pos & 7)))
00029       code += pos;
00030     pos += pos;
00031     (*bit_pos)++;
00032   }
00033   return code;
00034 }
00035
00036 int LZW_decode(ALLEGRO_FILE *file, ALGIF_BITMAP *bmp)
00037 {
00038   int orig_bit_size;
00039   char buf[256];
00040   int bit_size;
00041   int bit_pos;
00042   int clear_marker;
00043   int end_marker;
00044   struct
00045   {
00046     int prefix;
00047     int c;
00048     int len;
00049   } codes[4096]; /* Maximum bit size is 12. */
00050   int n;
00051   int i, prev, code, c;
00052   int out_pos = 0;
00053
00054   orig_bit_size = al_fgetc(file);
00055   n = 2 + (1 « orig_bit_size);
00056
00057   for (i = 0; i < n; i++)
00058   {
00059     codes[i].c = i;
00060     codes[i].len = 0;
00061   }
00062
00063   clear_marker = n - 2;
00064   end_marker = n - 1;
00065
00066   bit_size = orig_bit_size + 1;
00067
00068   bit_pos = 0;
00069
00070   /* Expect to read clear code as first code here. */
00071   prev = read_code(file, buf, &bit_pos, bit_size);
00072   // printf("%d %d = %d\n", bit_pos, bit_size, prev);
00073   if (prev == -1)
00074     return -1;
00075   do
00076   {
00077     code = read_code(file, buf, &bit_pos, bit_size);
00078     // printf("%d %d = %d\n", bit_pos, bit_size, code);
00079     if (code == -1)
00080       return -1;
00081     if (code == clear_marker)
00082     {
00083       bit_size = orig_bit_size;
00084       n = 1 « bit_size;
00085       n += 2;
```

```
00086       bit_size++;
00087       prev = code;
00088       continue;
00089     }
00090
00091     if (code == end_marker)
00092       break;
00093
00094     /* Known code: ok. Else: must be doubled char. */
00095     if (code < n)
00096       c = code;
00097     else
00098       c = prev;
00099
00100     /* Output the code. */
00101     out_pos += codes[c].len;
00102     i = 0;
00103     do
00104     {
00105       bmp->data[out_pos - i] = codes[c].c;
00106       if (codes[c].len)
00107         c = codes[c].prefix;
00108       else
00109         break;
00110       i++;
00111     } while (1);
00112
00113     out_pos++;
00114
00115     /* Unknown code -> must be double char. */
00116     if (code >= n)
00117     {
00118       bmp->data[out_pos] = codes[c].c;
00119       out_pos++;
00120     }
00121
00122     /* Except after clear marker, build new code. */
00123     if (prev != clear_marker)
00124     {
00125       codes[n].prefix = prev;
00126       codes[n].len = codes[prev].len + 1;
00127       codes[n].c = codes[c].c;
00128       n++;
00129     }
00130
00131     /* Out of bits? Increase. */
00132     if (n == (1 << bit_size))
00133     {
00134       if (bit_size < 12)
00135         bit_size++;
00136     }
00137
00138     prev = code;
00139   } while (1);
00140   return 0;
00141 }
```

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

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

Include dependency graph for allegro_stuff.c:



Data Structures

- struct allegro_t
- struct sounds_t

Macros

- #define FONT_HEIGHT 16
- #define SOUND_STREAM_FILE_CREDITS "credits_theme"
- #define SOUND_STREAM_FILE_MAIN "main_menu_theme"
- #define SOUND_STREAM_FILE_PAUSE "pause_menu_theme"
- #define SOUND_STREAM_FILE_PLAYING "playing_theme"
- #define SOUND_STREAM_FILE_RANKING "ranking_theme"
- #define SOUND_STREAM_FILE_RICK "rick"
- #define SOUND_STREAM_FILE_GAME_OVER "game_over"
- #define FONT_FILE_NAME "PublicPixel.ttf"
- #define SPRITE_HEART "minecraft_heart"
- #define SPRITE_BACKGROUND "sprite_background"
- #define SPRITE_CAR "sprite_cars"
- #define SPRITE_FROG "sprite_frog"
- #define SPRITE_LOG "sprite_log"
- #define SPRITE_TURTLES "sprite_turtles"
- #define SPRITE_MENU_HOME_BACK "sprite_menu_home_background"
- #define SPRITE_MENU_HOME "sprite_menu_home"
- #define SPRITE_MENU_DIFF_BACK "sprite_menu_diff_background"
- #define SPRITE_MENU_DIFF "sprite_menu_diff"
- #define SPRITE_MENU_PAUSE_BACK "sprite_menu_pause_background"
- #define SPRITE_MENU_PAUSE "sprite_menu_pause"
- #define SPRITE_MENU_GAME_OVER_BACK "sprite_menu_gameover_background"
- #define SPRITE_MENU_GAME_OVER "sprite_menu_gameover"
- #define SPRITE_CREDITS "sprite_credits"
- #define SPRITE_NAME "sprite_name"
- #define SPRITE_ICON "icon"
- #define SPRITE_DEAD "sprite_dead"
- #define SPRITE_BORDER "sprite_border"
- #define SPRITE_SPLASH "sprite_splash"
- #define SPRITE_COIN "sprite_coin"
- #define EXTENSION_SOUND_SAMPLE ".wav"
- #define EXTENSION_SOUND_STREAM ".opus"
- #define EXTENSION_SPRITES ".png"
- #define PATH_SOUND_STREAMS "../res/sounds/streams/"
- #define PATH_SOUND_SAMPLES "../res/sounds/samples/"
- #define PATH_FONTS "../res/fonts/"
- #define PATH_SPRITES "../res/sprites/"
- #define PATH_GIFS "../res/gifs/"
- #define GLOBAL_STREAM_VOLUME (double)0.5

Enumerations

- enum **SOUND_STREAM_STATES** { **SOUND_STREAM_STATE_NO_INIT** , **SOUND_STREAM_STATE_↩ INIT** , **SOUND_STREAM_STATE_PAUSE** , **SOUND_STREAM_STATE_PLAY** }

Functions

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

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

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

  Desinicializaciones de allegro.
- void allegro_reinit_display (void)

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

  Desinicializa el display.
- unsigned char allegro_get_last_key (void)

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

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

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

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

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

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

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

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

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

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

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

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

  Pone negro el display.
- void allegro_draw_background (void)

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

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

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

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

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

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

Selecciona musica de menu. Comienza pausada.

- void allegro_sound_set_stream_pause_menu (void)

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

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

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

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

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

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

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

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

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

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

  Reproduce efecto de bonus.
- void allegro_sound_play_effect_click (void)

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

  Reproduce efecto de choque.
- void allegro_sound_play_effect_drowned (void)

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

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

  Reproduce efecto de salto.
- void allegro_sound_play_effect_low_time (void)

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

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

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

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

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

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

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

  Dibuja un contorno rectangular.
- void allegro_rick_on (void)

- bool allegro_get_rick_flag (void)
- void allegro_set_rick_flag (bool state)
- void allegro_rick_off (void)
- void allegro_rick_draw (void)

Variables

- sprites_t sprites

4.24.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-10

Copyright

Copyright (c) 2022

Definition in file allegro_stuff.c.

4.24.2 Macro Definition Documentation

4.24.2.1 EXTENSION_SOUND_SAMPLE `#define EXTENSION_SOUND_SAMPLE ".wav"`

Definition at line 70 of file allegro_stuff.c.

4.24.2.2 EXTENSION_SOUND_STREAM `#define EXTENSION_SOUND_STREAM ".opus"`

Definition at line 71 of file allegro_stuff.c.

4.24.2.3 EXTENSION_SPRITES `#define EXTENSION_SPRITES ".png"`

Definition at line 72 of file allegro_stuff.c.

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

Definition at line 38 of file allegro_stuff.c.

4.24.2.5 FONT_HEIGHT `#define FONT_HEIGHT 16`

Definition at line 27 of file allegro_stuff.c.

4.24.2.6 GLOBAL_STREAM_VOLUME `#define GLOBAL_STREAM_VOLUME (double)0.5`

Definition at line 81 of file allegro_stuff.c.

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

Definition at line 77 of file allegro_stuff.c.

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

Definition at line 79 of file allegro_stuff.c.

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

Definition at line 76 of file allegro_stuff.c.

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

Definition at line 75 of file allegro_stuff.c.

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

Definition at line 78 of file allegro_stuff.c.

4.24.2.12 SOUND_STREAM_FILE_CREDITS `#define SOUND_STREAM_FILE_CREDITS "credits_theme"`

Definition at line 30 of file allegro_stuff.c.

4.24.2.13 SOUND_STREAM_FILE_GAME_OVER `#define SOUND_STREAM_FILE_GAME_OVER "game_over"`

Definition at line 36 of file allegro_stuff.c.

4.24.2.14 SOUND_STREAM_FILE_MAIN `#define SOUND_STREAM_FILE_MAIN "main_menu_theme"`

Definition at line 31 of file allegro_stuff.c.

4.24.2.15 SOUND_STREAM_FILE_PAUSE `#define SOUND_STREAM_FILE_PAUSE "pause_menu_theme"`

Definition at line 32 of file allegro_stuff.c.

4.24.2.16 SOUND_STREAM_FILE_PLAYING `#define SOUND_STREAM_FILE_PLAYING "playing_theme"`

Definition at line 33 of file allegro_stuff.c.

4.24.2.17 SOUND_STREAM_FILE_RANKING `#define SOUND_STREAM_FILE_RANKING "ranking_theme"`

Definition at line 34 of file allegro_stuff.c.

4.24.2.18 SOUND_STREAM_FILE_RICK `#define SOUND_STREAM_FILE_RICK "rick"`

Definition at line 35 of file allegro_stuff.c.

4.24.2.19 SPRITE_BACKGROUND `#define SPRITE_BACKGROUND "sprite_background"`

Definition at line 42 of file allegro_stuff.c.

4.24.2.20 SPRITE_BORDER `#define SPRITE_BORDER "sprite_border"`

Definition at line 65 of file allegro_stuff.c.

4.24.2.21 SPRITE_CAR `#define SPRITE_CAR "sprite_cars"`

Definition at line 43 of file allegro_stuff.c.

4.24.2.22 SPRITE_COIN `#define SPRITE_COIN "sprite_coin"`

Definition at line 67 of file allegro_stuff.c.

4.24.2.23 SPRITE_CREDITS `#define SPRITE_CREDITS "sprite_credits"`

Definition at line 61 of file allegro_stuff.c.

4.24.2.24 SPRITE_DEAD `#define SPRITE_DEAD "sprite_dead"`

Definition at line 64 of file allegro_stuff.c.

4.24.2.25 SPRITE_FROG `#define SPRITE_FROG "sprite_frog"`

Definition at line 44 of file allegro_stuff.c.

4.24.2.26 SPRITE_HEART `#define SPRITE_HEART "minecraft_heart"`

Definition at line 41 of file allegro_stuff.c.

4.24.2.27 SPRITE_ICON `#define SPRITE_ICON "icon"`

Definition at line 63 of file allegro_stuff.c.

4.24.2.28 SPRITE_LOG `#define SPRITE_LOG "sprite_log"`

Definition at line 45 of file allegro_stuff.c.

4.24.2.29 SPRITE_MENU_DIFF `#define SPRITE_MENU_DIFF "sprite_menu_diff"`

Definition at line 56 of file allegro_stuff.c.

4.24.2.30 SPRITE_MENU_DIFF_BACK `#define SPRITE_MENU_DIFF_BACK "sprite_menu_diff_background"`

Definition at line 55 of file allegro_stuff.c.

4.24.2.31 SPRITE_MENU_GAME_OVER `#define SPRITE_MENU_GAME_OVER "sprite_menu_gameover"`

Definition at line 60 of file allegro_stuff.c.

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

Definition at line 59 of file allegro_stuff.c.

4.24.2.33 SPRITE_MENU_HOME `#define SPRITE_MENU_HOME "sprite_menu_home"`

Definition at line 48 of file allegro_stuff.c.

4.24.2.34 SPRITE_MENU_HOME_BACK `#define SPRITE_MENU_HOME_BACK "sprite_menu_home_background"`

Definition at line 47 of file allegro_stuff.c.

4.24.2.35 SPRITE_MENU_PAUSE `#define SPRITE_MENU_PAUSE "sprite_menu_pause"`

Definition at line 58 of file allegro_stuff.c.

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

Definition at line 57 of file allegro_stuff.c.

4.24.2.37 SPRITE_NAME `#define SPRITE_NAME "sprite_name"`

Definition at line 62 of file allegro_stuff.c.

4.24.2.38 SPRITE_SPLASH `#define SPRITE_SPLASH "sprite_splash"`

Definition at line 66 of file allegro_stuff.c.

4.24.2.39 SPRITE_TURTLES `#define SPRITE_TURTLES "sprite_turtles"`

Definition at line 46 of file allegro_stuff.c.

4.24.3 Enumeration Type Documentation

4.24.3.1 SOUND_STREAM_STATES `enum SOUND_STREAM_STATES`

Definition at line 137 of file allegro_stuff.c.

4.24.4 Function Documentation

4.24.4.1 allegro_clear_display() `void allegro_clear_display ( void )`

Pone negro el display.

Definition at line 456 of file allegro_stuff.c.

4.24.4.2 allegro_deinit_display() `void allegro_deinit_display (`

```
void )
```

Desinicializa el display.

Definition at line 380 of file allegro_stuff.c.

4.24.4.3 allegro_deinits() `void allegro_deinits (`

```
void )
```

Desinicializaciones de allegro.

Definition at line 334 of file allegro_stuff.c.

4.24.4.4 allegro_draw_background() `void allegro_draw_background (`

```
void )
```

Dibuja la imagen de fondo.

Definition at line 461 of file allegro_stuff.c.

4.24.4.5 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 736 of file allegro_stuff.c.

4.24.4.6 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 466 of file allegro_stuff.c.

4.24.4.7 allegro_get_event_queue() `ALLEGRO_EVENT_QUEUE * allegro_get_event_queue ( void )`

Devuelve puntero a la cola de eventos.

Definition at line 476 of file allegro_stuff.c.

4.24.4.8 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 389 of file allegro_stuff.c.

4.24.4.9 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 406 of file allegro_stuff.c.

4.24.4.10 allegro_get_rick_flag() `bool allegro_get_rick_flag ( void )`

Definition at line 754 of file allegro_stuff.c.

4.24.4.11 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 421 of file allegro_stuff.c.

4.24.4.12 allegro_get_var_event() `ALLEGRO_EVENT allegro_get_var_event ( void )`

Devuelve el evento de allegro.

Returns

ALLEGRO_EVENT

Definition at line 416 of file allegro_stuff.c.

4.24.4.13 allegro_get_var_font() `ALLEGRO_FONT * allegro_get_var_font ( void )`

Devuelve la fuente de allegro.

Returns

ALLEGRO_FONT

Definition at line 441 of file allegro_stuff.c.

4.24.4.14 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 446 of file allegro_stuff.c.

4.24.4.15 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 451 of file allegro_stuff.c.

4.24.4.16 allegro_get_var_redraw() `bool allegro_get_var_redraw ( void )`

Devuelve flag de renderización.

Returns

true Renderiza

false No renderiza

Definition at line 426 of file allegro_stuff.c.

4.24.4.17 allegro_inits() `void allegro_inits ( void )`

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

Definition at line 283 of file allegro_stuff.c.

4.24.4.18 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 471 of file allegro_stuff.c.

4.24.4.19 allegro_reinit_display() `void allegro_reinit_display (`

`void )`

Reinicializa el display de allegro.

Definition at line 347 of file allegro_stuff.c.

4.24.4.20 allegro_rick_draw() `void allegro_rick_draw (`

`void )`

Definition at line 771 of file allegro_stuff.c.

4.24.4.21 allegro_rick_off() `void allegro_rick_off (`

`void )`

Definition at line 764 of file allegro_stuff.c.

4.24.4.22 allegro_rick_on() `void allegro_rick_on (`

`void )`

Definition at line 746 of file allegro_stuff.c.

4.24.4.23 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 394 of file allegro_stuff.c.

4.24.4.24 allegro_set_rick_flag() `void allegro_set_rick_flag (`

`bool` *`state`* `)`

Parameters

state	

Definition at line 759 of file allegro_stuff.c.

4.24.4.25 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 431 of file allegro_stuff.c.

4.24.4.26 allegro_set_var_event() void allegro_set_var_event (ALLEGRO_EVENT *event*)

Carga un evento de allegro.

Parameters

event	

Definition at line 481 of file allegro_stuff.c.

4.24.4.27 allegro_set_var_redraw() void allegro_set_var_redraw (bool *state*)

Setea flag de renderizacion.

Parameters

state	true or false

Definition at line 436 of file allegro_stuff.c.

4.24.4.28 allegro_sound_pause_stream() void allegro_sound_pause_stream (void)

Pausa la musica actual, si hay alguna seleccionada.

Definition at line 629 of file allegro_stuff.c.

4.24.4.29 allegro_sound_play_effect_bonus() void allegro_sound_play_effect_bonus (void)

Reproduce efecto de bonus.

Definition at line 667 of file allegro_stuff.c.

4.24.4.30 allegro_sound_play_effect_click() void allegro_sound_play_effect_click (void)

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

Definition at line 672 of file allegro_stuff.c.

4.24.4.31 allegro_sound_play_effect_coin_drop() void allegro_sound_play_effect_coin_drop (void)

Reproduce efecto de moneda tirada

Definition at line 727 of file allegro_stuff.c.

4.24.4.32 allegro_sound_play_effect_crash() void allegro_sound_play_effect_crash (void)

Reproduce efecto de choque.

Definition at line 677 of file allegro_stuff.c.

4.24.4.33 allegro_sound_play_effect_drowned() void allegro_sound_play_effect_drowned (void)

Reproduce efecto de ahogado (toco agua)

Definition at line 682 of file allegro_stuff.c.

4.24.4.34 allegro_sound_play_effect_exiting() void allegro_sound_play_effect_exiting (void)

Reproduce efecto de 'saliendo'.

Definition at line 717 of file allegro_stuff.c.

4.24.4.35 allegro_sound_play_effect_goal() void allegro_sound_play_effect_goal (void)

Reproduce efecto de 'llego a la meta'.

Definition at line 687 of file allegro_stuff.c.

4.24.4.36 allegro_sound_play_effect_jump() void allegro_sound_play_effect_jump (void)

Reproduce efecto de salto.

Definition at line 692 of file allegro_stuff.c.

4.24.4.37 allegro_sound_play_effect_low_time() void allegro_sound_play_effect_low_time (void)

Reproduce efecto de 'queda poco tiempo'.

Definition at line 697 of file allegro_stuff.c.

4.24.4.38 allegro_sound_play_effect_menu_enter() void allegro_sound_play_effect_menu_enter (void)

Reproduce efecto de 'menu enter'.

Definition at line 707 of file allegro_stuff.c.

4.24.4.39 allegro_sound_play_effect_new_max_score() void allegro_sound_play_effect_new_max_score (void)

Reproduce efecto de 'new_max_score'.

Definition at line 712 of file allegro_stuff.c.

4.24.4.40 allegro_sound_play_effect_no_time() void allegro_sound_play_effect_no_time (void)

Reproduce efecto de 'sin tiempo'.

Definition at line 722 of file allegro_stuff.c.

4.24.4.41 allegro_sound_play_effect_run_completed() `void allegro_sound_play_effect_run_completed (`

```
            void  )
```

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

Definition at line 702 of file allegro_stuff.c.

4.24.4.42 allegro_sound_play_stream() `void allegro_sound_play_stream (`

```
            void  )
```

Reproduce la musica actual, si hay alguna seleccionada.

Definition at line 620 of file allegro_stuff.c.

4.24.4.43 allegro_sound_restart_stream() `void allegro_sound_restart_stream (`

```
            void  )
```

Reinicia la musica actual, si hay alguna seleccionada.

Definition at line 638 of file allegro_stuff.c.

4.24.4.44 allegro_sound_set_stream_credits() `void allegro_sound_set_stream_credits (`

```
            void  )
```

Selecciona musica de creditos. Comienza pausada.

Definition at line 489 of file allegro_stuff.c.

4.24.4.45 allegro_sound_set_stream_gain_down() `void allegro_sound_set_stream_gain_down (`

```
            void  )
```

Reduce en 0.1 la ganancia de stream.

Definition at line 655 of file allegro_stuff.c.

4.24.4.46 allegro_sound_set_stream_gain_up() `void allegro_sound_set_stream_gain_up (`

```
            void  )
```

Aumenta en 0.1 la ganancia de stream.

Definition at line 646 of file allegro_stuff.c.

4.24.4.47 allegro_sound_set_stream_game_over() `void allegro_sound_set_stream_game_over ( void )`

Selecciona musica de game over. Comienza pausada.

Definition at line 585 of file allegro_stuff.c.

4.24.4.48 allegro_sound_set_stream_main_menu() `void allegro_sound_set_stream_main_menu ( void )`

Selecciona musica de menu. Comienza pausada.

Definition at line 505 of file allegro_stuff.c.

4.24.4.49 allegro_sound_set_stream_pause_menu() `void allegro_sound_set_stream_pause_menu ( void )`

Selecciona musica de pausa. Comienza pausada.

Definition at line 521 of file allegro_stuff.c.

4.24.4.50 allegro_sound_set_stream_playing() `void allegro_sound_set_stream_playing ( void )`

Selecciona musica de jugando. Comienza pausada.

Definition at line 553 of file allegro_stuff.c.

4.24.4.51 allegro_sound_set_stream_ranking() `void allegro_sound_set_stream_ranking ( void )`

Selecciona musica de ranking. Comienza pausada.

Definition at line 537 of file allegro_stuff.c.

4.24.4.52 allegro_sound_set_stream_rick() `void allegro_sound_set_stream_rick ( void )`

Definition at line 569 of file allegro_stuff.c.

4.24.4.53 allegro_sound_toggle_stream() `void allegro_sound_toggle_stream ( void )`

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

Definition at line 604 of file allegro_stuff.c.

4.24.4.54 allegro_wait_for_event() `ALLEGRO_EVENT_TYPE allegro_wait_for_event ( void )`

Espera a que ocurra un evento y lo devuelve.

Returns

ALLEGRO_EVENT_TYPE

Definition at line 399 of file allegro_stuff.c.

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

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

Parameters

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

Definition at line 274 of file allegro_stuff.c.

4.24.5 Variable Documentation

4.24.5.1 sprites `sprites_t sprites`

Definition at line 150 of file allegro_stuff.c.

4.25 allegro_stuff.c

Go to the documentation of this file.

```
00001
00012 /*****************************************************************************
00013  * INCLUDE HEADER FILES
```

```
00014  ******************************************************************************/
00015
00016 #include "allegro_stuff.h"
00017 #include "geometry.h"
00018 #include <string.h>
00019
00020 #include "./algif5/algif.h"
00021
00022 /*******************************************************************************
00023  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00024  ******************************************************************************/
00025
00026 // Altura de la fuente
00027 #define FONT_HEIGHT 16
00028
00029 // Nombres de los stream files
00030 #define SOUND_STREAM_FILE_CREDITS "credits_theme"
00031 #define SOUND_STREAM_FILE_MAIN "main_menu_theme"
00032 #define SOUND_STREAM_FILE_PAUSE "pause_menu_theme"
00033 #define SOUND_STREAM_FILE_PLAYING "playing_theme"
00034 #define SOUND_STREAM_FILE_RANKING "ranking_theme"
00035 #define SOUND_STREAM_FILE_RICK "rick"
00036 #define SOUND_STREAM_FILE_GAME_OVER "game_over"
00037
00038 #define FONT_FILE_NAME "PublicPixel.ttf"
00039
00040 // Nombres de los sprites
00041 #define SPRITE_HEART "minecraft_heart"
00042 #define SPRITE_BACKGROUND "sprite_background"
00043 #define SPRITE_CAR "sprite_cars"
00044 #define SPRITE_FROG "sprite_frog"
00045 #define SPRITE_LOG "sprite_log"
00046 #define SPRITE_TURTLES "sprite_turtles"
00047 #define SPRITE_MENU_HOME_BACK "sprite_menu_home_background"
00048 #define SPRITE_MENU_HOME "sprite_menu_home"
00049 /*
00050 #define SPRITE_MENU_DIFF_BACK       "sprite_menu_home_background"
00051 #define SPRITE_MENU_DIFF            "sprite_menu_home"
00052 #define SPRITE_MENU_PAUSE_BACK      "sprite_menu_home_background"
00053 #define SPRITE_MENU_PAUSE           "sprite_menu_home"
00054 */
00055 #define SPRITE_MENU_DIFF_BACK "sprite_menu_diff_background"
00056 #define SPRITE_MENU_DIFF "sprite_menu_diff"
00057 #define SPRITE_MENU_PAUSE_BACK "sprite_menu_pause_background"
00058 #define SPRITE_MENU_PAUSE "sprite_menu_pause"
00059 #define SPRITE_MENU_GAME_OVER_BACK "sprite_menu_gameover_background"
00060 #define SPRITE_MENU_GAME_OVER "sprite_menu_gameover"
00061 #define SPRITE_CREDITS "sprite_credits"
00062 #define SPRITE_NAME "sprite_name"
00063 #define SPRITE_ICON "icon"
00064 #define SPRITE_DEAD "sprite_dead"
00065 #define SPRITE_BORDER "sprite_border"
00066 #define SPRITE_SPLASH "sprite_splash"
00067 #define SPRITE_COIN "sprite_coin"
00068
00069 // Extensiones
00070 #define EXTENSION_SOUND_SAMPLE ".wav"
00071 #define EXTENSION_SOUND_STREAM ".opus"
00072 #define EXTENSION_SPRITES ".png"
00073
00074 // Local paths
00075 #define PATH_SOUND_STREAMS "../res/sounds/streams/"
00076 #define PATH_SOUND_SAMPLES "../res/sounds/samples/"
00077 #define PATH_FONTS "../res/fonts/"
00078 #define PATH_SPRITES "../res/sprites/"
00079 #define PATH_GIFS "../res/gifs/"
00080
00081 #define GLOBAL_STREAM_VOLUME (double)0.5
00082
00083 /*******************************************************************************
00084  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00085  ******************************************************************************/
00086
00087 typedef struct
00088 {
00090   ALLEGRO_TIMER *timer;
00091
00092   // cola de eventos
00093   ALLEGRO_EVENT_QUEUE *queue;
00094
00095   // display
00096   ALLEGRO_DISPLAY *disp;
00097
00098   // fuente builtin
00099   ALLEGRO_FONT *font;
00100   int font_h; // altura de un caracter
00101   int font_w; // ancho de un caracter
```

```
00102
00103   // variable evento
00104   ALLEGRO_EVENT event;
00105
00106   // flag para salir el programa
00107   bool done;
00108   // flag para renderizar
00109   bool redraw;
00110
00111 } allegro_t;
00112
00113 typedef struct
00114 {
00115   ALLEGRO_AUDIO_STREAM *stream;
00116   unsigned char stream_state;
00117
00118   struct
00119   {
00120     ALLEGRO_SAMPLE *jump;
00121     ALLEGRO_SAMPLE *crash;
00122     ALLEGRO_SAMPLE *goal;
00123     ALLEGRO_SAMPLE *low_time;
00124     ALLEGRO_SAMPLE *click;
00125     ALLEGRO_SAMPLE *bonus;
00126     ALLEGRO_SAMPLE *run_completed;
00127     ALLEGRO_SAMPLE *drowned;
00128     ALLEGRO_SAMPLE *menu_enter;
00129     ALLEGRO_SAMPLE *new_max_score;
00130     ALLEGRO_SAMPLE *exiting;
00131     ALLEGRO_SAMPLE *no_time;
00132     ALLEGRO_SAMPLE *coin_drop;
00133   } samples;
00134
00135 } sounds_t;
00136
00137 enum SOUND_STREAM_STATES
00138 {
00139   SOUND_STREAM_STATE_NO_INIT,
00140   SOUND_STREAM_STATE_INIT,
00141   SOUND_STREAM_STATE_PAUSE,
00142   SOUND_STREAM_STATE_PLAY
00143 };
00144
00145 /*****************************************************************************
00146  * VARIABLES WITH GLOBAL SCOPE
00147  *****************************************************************************/
00148
00149 // estructura con punteros a sprites
00150 sprites_t sprites;
00151
00152 /*****************************************************************************
00153  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00154  *****************************************************************************/
00155
00167 static ALLEGRO_BITMAP *sprite_cut(ALLEGRO_BITMAP *source_bmp, int x, int y, int w, int h);
00168
00173 static void sprites_init(void);
00174
00190 static char *make_sprite_path(char *file_name, char *prev_str);
00191
00196 static void sprites_deinit(void);
00197
00202 static void audio_init(void);
00203
00208 static void audio_deinit(void);
00209
00218 static bool init_audio_stream(const char *file, float gain);
00219
00228 static bool init_sample(ALLEGRO_SAMPLE **sample, const char *file);
00229
00234 static void rick_init(void);
00235
00240 static void rick_deinit(void);
00241
00242 /*****************************************************************************
00243  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00244  *****************************************************************************/
00245
00246 // variables principales de allegro
00247 static allegro_t allegro_vars;
00248
00249 // Ultima tecla presionada
00250 static unsigned char last_key;
00251
00252 // variable con los sonidos/musicas del juego
00253 static sounds_t sounds;
00254
```

```
00255 // nombre del ultimo stream inicializado
00256 static char last_init_stream[30];
00257
00258 static ALGIF_ANIMATION *rick;
00259
00260 static char rick_prev_stream[30];
00261
00262 static bool rick_flag;
00263
00264 static ALLEGRO_MONITOR_INFO monitor_info;
00265
00266 static double stream_gain = GLOBAL_STREAM_VOLUME;
00267
00268 /*****************************************************************************
00269  *****************************************************************************
00270             GLOBAL FUNCTION DEFINITIONS
00271  *****************************************************************************
00272  *****************************************************************************/
00273
00274 void must_init(bool test, const char *description)
00275 {
00276   if (!test)
00277   {
00278     printf("~no se pudo inicializar %s~\n", description);
00279     exit(EXIT_FAILURE);
00280   }
00281 }
00282
00283 void allegro_inits(void)
00284 {
00285   must_init(al_init(), "allegro");
00286   must_init(al_install_keyboard(), "keyboard");
00287   must_init(al_install_mouse(), "mouse");
00288   must_init(al_init_image_addon(), "image");
00289   al_init_font_addon();
00290   must_init(al_init_ttf_addon(), "ttf addon");
00291
00292   // timer que actualiza cada 1/60 segundos (60fps)
00293   allegro_vars.timer = al_create_timer(1.0 / FPS);
00294   must_init(allegro_vars.timer, "timer");
00295
00296   // cola de eventos
00297   allegro_vars.queue = al_create_event_queue();
00298   must_init(allegro_vars.queue, "queue");
00299
00300   // Inicializa los spritesheets.
00301   sprites_init();
00302
00303   // para dibujar figuras primitivas (círculos, rectángulos, líneas, rellenos o no, etc.)
00304   must_init(al_init_primitives_addon(), "primitives");
00305
00306   // registra eventos posibles
00307   al_register_event_source(allegro_vars.queue, al_get_keyboard_event_source());
00308   al_register_event_source(allegro_vars.queue, al_get_timer_event_source(allegro_vars.timer));
00309   al_register_event_source(allegro_vars.queue, al_get_mouse_event_source());
00310
00311   // flag para salir el programa
00312   allegro_vars.done = false;
00313   // flag para renderizar
00314   allegro_vars.redraw = false;
00315
00316   // audio
00317   must_init(al_install_audio(), "audio");
00318   must_init(al_init_acodec_addon(), "audio codecs");
00319   must_init(al_reserve_samples(16), "reserve samples");
00320
00321   audio_init();
00322
00323   rick_init();
00324
00325   must_init(al_get_monitor_info(0, &monitor_info), "getting monitor info");
00326
00327   // creacion del display
00328   allegro_reinit_display();
00329
00330   // inicializa timer
00331   al_start_timer(allegro_vars.timer);
00332 }
00333
00334 void allegro_deinits(void)
00335 {
00336   rick_deinit();
00337   sprites_deinit();
00338   audio_deinit();
00339   al_destroy_font(allegro_vars.font);
00340
00341     //al_destroy_display(allegro_vars.disp);
```

```
00342
00343   al_destroy_timer(allegro_vars.timer);
00344   al_destroy_event_queue(allegro_vars.queue);
00345 }
00346
00347 void allegro_reinit_display(void)
00348 {
00349
00350   al_set_new_bitmap_flags(ALLEGRO_MIN_LINEAR | ALLEGRO_MAG_LINEAR);
00351   // Para tener aceleracion por HW desde la GPU (hace que no explote con los draw_text)
00352   al_set_new_bitmap_flags(ALLEGRO_VIDEO_BITMAP);
00353   // al_set_new_display_flags(ALLEGRO_RESIZABLE);
00354
00355   // Titulo de la ventana
00356   al_set_new_window_title("~ Programación I ~ TP Final ~ Frogger ~");
00357   // Centrado en pantalla, según el monitor
00358   al_set_new_window_position(monitor_info.x2 / 2 - DISPLAY_W / 2, monitor_info.y2 / 2 - DISPLAY_H / 2
      - 50);
00359
00360   // opciones para el display (antialiasing)
00361   al_set_new_display_option(ALLEGRO_SAMPLE_BUFFERS, 1, ALLEGRO_SUGGEST);
00362   al_set_new_display_option(ALLEGRO_SAMPLES, 8, ALLEGRO_SUGGEST);
00363
00364   // creación del display
00365   allegro_vars.disp = al_create_display(DISPLAY_W, DISPLAY_H);
00366   must_init(allegro_vars.disp, "display");
00367   al_register_event_source(allegro_vars.queue, al_get_display_event_source(allegro_vars.disp));
00368
00369   al_set_display_icon(allegro_vars.disp, sprites.icon);
00370
00371   // Reload de la fuente
00372   char string[60] = PATH_FONTS;
00373   strcat(string, FONT_FILE_NAME);
00374   allegro_vars.font = al_load_font(string, FONT_HEIGHT, 0);
00375   must_init(allegro_vars.font, "font");
00376   allegro_vars.font_h = al_get_font_line_height(allegro_vars.font);
00377   allegro_vars.font_w = al_get_text_width(allegro_vars.font, "a");
00378 }
00379
00380 void allegro_deinit_display(void)
00381 {
00382   if (allegro_vars.disp != NULL)
00383   {
00384     al_unregister_event_source(allegro_vars.queue, al_get_display_event_source(allegro_vars.disp));
00385     al_destroy_display(allegro_vars.disp);
00386   }
00387 }
00388
00389 unsigned char allegro_get_last_key(void)
00390 {
00391   return (last_key);
00392 }
00393
00394 void allegro_set_last_key(unsigned char allegro_key_code)
00395 {
00396   last_key = allegro_key_code;
00397 }
00398
00399 ALLEGRO_EVENT_TYPE allegro_wait_for_event(void)
00400 {
00401   al_wait_for_event(allegro_vars.queue, &allegro_vars.event);
00402
00403   return (allegro_vars.event.type);
00404 }
00405
00406 ALLEGRO_EVENT *allegro_get_next_event(void)
00407 {
00408   bool flag = al_get_next_event(allegro_vars.queue, &allegro_vars.event);
00409
00410   if (flag)
00411     return (&allegro_vars.event);
00412   else
00413     return NULL;
00414 }
00415
00416 ALLEGRO_EVENT allegro_get_var_event(void)
00417 {
00418   return (allegro_vars.event);
00419 }
00420
00421 bool allegro_get_var_done(void)
00422 {
00423   return (allegro_vars.done);
00424 }
00425
00426 bool allegro_get_var_redraw(void)
00427 {
```

```
00428   return (allegro_vars.redraw);
00429 }
00430
00431 void allegro_set_var_done(bool state)
00432 {
00433   allegro_vars.done = state;
00434 }
00435
00436 void allegro_set_var_redraw(bool state)
00437 {
00438   allegro_vars.redraw = state;
00439 }
00440
00441 ALLEGRO_FONT *allegro_get_var_font(void)
00442 {
00443   return (allegro_vars.font);
00444 }
00445
00446 int allegro_get_var_font_h(void)
00447 {
00448   return (allegro_vars.font_h);
00449 }
00450
00451 int allegro_get_var_font_w(void)
00452 {
00453   return (allegro_vars.font_w);
00454 }
00455
00456 void allegro_clear_display(void)
00457 {
00458   al_clear_to_color(al_map_rgb(0, 0, 0));
00459 }
00460
00461 void allegro_draw_background(void)
00462 {
00463   al_draw_bitmap(sprites.background, 0, 0, 0);
00464 }
00465
00466 void allegro_draw_menu_background(int window)
00467 {
00468   al_draw_bitmap(sprites.menu[window].background, 0, 0, 0);
00469 }
00470
00471 bool allegro_is_event_queueVacia(void)
00472 {
00473   return (al_is_event_queue_empty(allegro_vars.queue));
00474 }
00475
00476 ALLEGRO_EVENT_QUEUE *allegro_get_event_queue(void)
00477 {
00478   return (allegro_vars.queue);
00479 }
00480
00481 void allegro_set_var_event(ALLEGRO_EVENT event)
00482 {
00483   allegro_vars.event = event;
00484 }
00485
00486 #pragma region allegro_sound
00487
00488 #pragma region allegro_sound_set_stream
00489 void allegro_sound_set_stream_credits(void)
00490 {
00491   char file[] = SOUND_STREAM_FILE_CREDITS;
00492
00493   // si ya estaba inicializado...
00494   if (strcmp(file, last_init_stream) == 0)
00495   {
00496     allegro_sound_pause_stream();
00497   }
00498   else
00499   {
00500     must_init(init_audio_stream(file, stream_gain),
00501               "credits stream");
00502   }
00503 }
00504
00505 void allegro_sound_set_stream_main_menu(void)
00506 {
00507   char file[] = SOUND_STREAM_FILE_MAIN;
00508
00509   // si ya estaba inicializado...
00510   if (strcmp(file, last_init_stream) == 0)
00511   {
00512     allegro_sound_pause_stream();
00513   }
00514   else
```

```
00515   {
00516     must_init(init_audio_stream(file, stream_gain),
00517               "main menu stream");
00518   }
00519 }
00520
00521 void allegro_sound_set_stream_pause_menu(void)
00522 {
00523   char file[] = SOUND_STREAM_FILE_PAUSE;
00524
00525   // si ya estaba inicializado...
00526   if (strcmp(file, last_init_stream) == 0)
00527   {
00528     allegro_sound_pause_stream();
00529   }
00530   else
00531   {
00532     must_init(init_audio_stream(file, stream_gain),
00533               "pause menu stream");
00534   }
00535 }
00536
00537 void allegro_sound_set_stream_ranking(void)
00538 {
00539   char file[] = SOUND_STREAM_FILE_RANKING;
00540
00541   // si ya estaba inicializado...
00542   if (strcmp(file, last_init_stream) == 0)
00543   {
00544     allegro_sound_pause_stream();
00545   }
00546   else
00547   {
00548     must_init(init_audio_stream(file, stream_gain),
00549               "ranking stream");
00550   }
00551 }
00552
00553 void allegro_sound_set_stream_playing(void)
00554 {
00555   char file[] = SOUND_STREAM_FILE_PLAYING;
00556
00557   // si ya estaba inicializado...
00558   if (strcmp(file, last_init_stream) == 0)
00559   {
00560     allegro_sound_pause_stream();
00561   }
00562   else
00563   {
00564     must_init(init_audio_stream(file, stream_gain),
00565               "playing stream");
00566   }
00567 }
00568
00569 void allegro_sound_set_stream_rick(void)
00570 {
00571   char file[] = SOUND_STREAM_FILE_RICK;
00572
00573   // si ya estaba inicializado...
00574   if (strcmp(file, last_init_stream) == 0)
00575   {
00576     allegro_sound_pause_stream();
00577   }
00578   else
00579   {
00580     must_init(init_audio_stream(file, stream_gain),
00581               "credtis stream");
00582   }
00583 }
00584
00585 void allegro_sound_set_stream_game_over(void)
00586 {
00587   char file[] = SOUND_STREAM_FILE_GAME_OVER;
00588
00589   // si ya estaba inicializado...
00590   if (strcmp(file, last_init_stream) == 0)
00591   {
00592     allegro_sound_pause_stream();
00593   }
00594   else
00595   {
00596     must_init(init_audio_stream(file, stream_gain),
00597               "game over stream");
00598   }
00599 }
00600
00601 #pragma endregion allegro_sound_set_stream
```

```
00602
00603 #pragma region allegro_sound_control
00604 void allegro_sound_toggle_stream(void)
00605 {
00606   if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00607   {
00608     bool state = al_get_audio_stream_playing(sounds.stream);
00609
00610     must_init(al_set_audio_stream_playing(sounds.stream, !state),
00611               "set to toggle stream");
00612
00613     if (!state)
00614       sounds.stream_state = SOUND_STREAM_STATE_PAUSE;
00615     else
00616       sounds.stream_state = SOUND_STREAM_STATE_PLAY;
00617   }
00618 }
00619
00620 void allegro_sound_play_stream(void)
00621 {
00622   if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00623   {
00624     al_set_audio_stream_playing(sounds.stream, true);
00625     sounds.stream_state = SOUND_STREAM_STATE_PLAY;
00626   }
00627 }
00628
00629 void allegro_sound_pause_stream(void)
00630 {
00631   if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00632   {
00633     al_set_audio_stream_playing(sounds.stream, false);
00634     sounds.stream_state = SOUND_STREAM_STATE_PAUSE;
00635   }
00636 }
00637
00638 void allegro_sound_restart_stream(void)
00639 {
00640   if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
00641   {
00642     init_audio_stream(last_init_stream, stream_gain);
00643   }
00644 }
00645
00646 void allegro_sound_set_stream_gain_up(void)
00647 {
00648   if (stream_gain <= 0.9)
00649   {
00650     stream_gain += 0.1;
00651     al_set_audio_stream_gain(sounds.stream, stream_gain);
00652   }
00653 }
00654
00655 void allegro_sound_set_stream_gain_down(void)
00656 {
00657   if (stream_gain >= 0.1)
00658   {
00659     stream_gain -= 0.1;
00660     al_set_audio_stream_gain(sounds.stream, stream_gain);
00661   }
00662 }
00663
00664 #pragma endregion allegro_sound_control
00665
00666 #pragma region allegro_sound_play_sample
00667 void allegro_sound_play_effect_bonus(void)
00668 {
00669   al_play_sample(sounds.samples.bonus, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00670 }
00671
00672 void allegro_sound_play_effect_click(void)
00673 {
00674   al_play_sample(sounds.samples.click, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00675 }
00676
00677 void allegro_sound_play_effect_crash(void)
00678 {
00679   al_play_sample(sounds.samples.crash, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00680 }
00681
00682 void allegro_sound_play_effect_drowned(void)
00683 {
00684   al_play_sample(sounds.samples.drowned, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00685 }
00686
00687 void allegro_sound_play_effect_goal(void)
00688 {
```

```
00689   al_play_sample(sounds.samples.goal, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00690 }
00691
00692 void allegro_sound_play_effect_jump(void)
00693 {
00694   al_play_sample(sounds.samples.jump, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00695 }
00696
00697 void allegro_sound_play_effect_low_time(void)
00698 {
00699   al_play_sample(sounds.samples.low_time, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00700 }
00701
00702 void allegro_sound_play_effect_run_completed(void)
00703 {
00704   al_play_sample(sounds.samples.run_completed, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00705 }
00706
00707 void allegro_sound_play_effect_menu_enter(void)
00708 {
00709   al_play_sample(sounds.samples.menu_enter, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00710 }
00711
00712 void allegro_sound_play_effect_new_max_score(void)
00713 {
00714   al_play_sample(sounds.samples.new_max_score, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00715 }
00716
00717 void allegro_sound_play_effect_exiting(void)
00718 {
00719   al_play_sample(sounds.samples.exiting, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00720 }
00721
00722 void allegro_sound_play_effect_no_time(void)
00723 {
00724   al_play_sample(sounds.samples.no_time, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00725 }
00726
00727 void allegro_sound_play_effect_coin_drop(void)
00728 {
00729   al_play_sample(sounds.samples.coin_drop, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, 0);
00730 }
00731
00732 #pragma endregion allegro_sound_play_sample
00733
00734 #pragma endregion allegro_sound
00735
00736 void allegro_draw_hitbox(int x, int y, int w, int h)
00737 {
00738   al_draw_rectangle(x,
00739                     y,
00740                     x + w,
00741                     y + h,
00742                     al_map_rgb(100, 100, 100),
00743                     1); // grosor
00744 }
00745
00746 void allegro_rick_on(void)
00747 {
00748   strcpy(rick_prev_stream, last_init_stream);
00749
00750   allegro_sound_set_stream_rick();
00751   allegro_sound_play_stream();
00752 }
00753
00754 bool allegro_get_rick_flag(void)
00755 {
00756   return rick_flag;
00757 }
00758
00759 void allegro_set_rick_flag(bool state)
00760 {
00761   rick_flag = state;
00762 }
00763
00764 void allegro_rick_off(void)
00765 {
00766   must_init(init_audio_stream(rick_prev_stream, stream_gain),
00767             "retornando stream ~~ sacando rick");
00768   allegro_sound_play_stream();
00769 }
00770
00771 void allegro_rick_draw(void)
00772 {
00773   al_draw_bitmap(algif_get_bitmap(rick, al_get_time()), 100, DISPLAY_H / 2, 0);
00774 }
00775
```

```
00776 /*****************************************************************************
00777  *****************************************************************************
00778             LOCAL FUNCTION DEFINITIONS
00779  *****************************************************************************
00780  *****************************************************************************/
00781
00782 static ALLEGRO_BITMAP *sprite_cut(ALLEGRO_BITMAP *source_bmp, int x, int y, int w, int h)
00783 {
00784   ALLEGRO_BITMAP *sprite = al_create_sub_bitmap(source_bmp, x, y, w, h);
00785   must_init(sprite, "sprite cut");
00786   return sprite;
00787 }
00788
00789 static void sprites_init(void)
00790 {
00791   int i, offset;
00792   pair_xy_t temp_xy;
00793   int temp_w, temp_h;
00794
00795   char *path = NULL;
00796
00797   // de la rana completo
00798   path = make_sprite_path(SPRITE_FROG, NULL);
00799   sprites.frog_uncut = al_load_bitmap(path);
00800
00801   // se particiona el de la rana en sus 8 partes
00802   for (i = 0; i < FROG_FRAMES; i++)
00803   {
00804     temp_xy = getXYFromFrogFrame(i);
00805
00806     if (!(i % 2)) // los sprites pares
00807     {
00808       temp_w = SPRITE_SIZE_FROG_STATIC_W;
00809       temp_h = SPRITE_SIZE_FROG_STATIC_H;
00810     }
00811     else if (i == 1 || i == 7)
00812     {
00813       temp_w = SPRITE_SIZE_FROG_DYNAMIC_SHORT;
00814       temp_h = SPRITE_SIZE_FROG_DYNAMIC_LONG;
00815     }
00816     else
00817     {
00818       temp_w = SPRITE_SIZE_FROG_DYNAMIC_LONG;
00819       temp_h = SPRITE_SIZE_FROG_DYNAMIC_SHORT;
00820     }
00821
00822     sprites.frog[i] = sprite_cut(sprites.frog_uncut, temp_xy.x, temp_xy.y, temp_w, temp_h);
00823   }
00824
00825   // el del fondo
00826   path = make_sprite_path(SPRITE_BACKGROUND, path);
00827   sprites.background = al_load_bitmap(path);
00828
00829   // el de los troncos
00830   path = make_sprite_path(SPRITE_LOG, path);
00831   sprites.log = al_load_bitmap(path);
00832
00833   // recorte de los autos.
00834
00835   path = make_sprite_path(SPRITE_CAR, path);
00836   sprites.cars_uncut = al_load_bitmap(path);
00837
00838   sprites.car[0] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(0).x, getXYFromCarFrame(0).y,
      CAR_W, CAR_H);
00839   sprites.car[1] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(1).x, getXYFromCarFrame(1).y,
      CAR_W, CAR_H);
00840   sprites.car[2] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(2).x, getXYFromCarFrame(2).y,
      CAR_W, CAR_H);
00841   sprites.car[3] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(3).x, getXYFromCarFrame(3).y,
      CAR_TRUCK_FIRE_W, CAR_H);
00842   sprites.car[4] = sprite_cut(sprites.cars_uncut, getXYFromCarFrame(4).x, getXYFromCarFrame(4).y,
      CAR_TRUCK_W, CAR_H);
00843
00844   // el de las tortugas sin recortar
00845   path = make_sprite_path(SPRITE_TURTLES, path);
00846   sprites.turtle_uncut = al_load_bitmap(path);
00847
00848   // se recortan los de la tortuga en sus 11 partes
00849   for (i = 0; i < TURTLE_FRAMES; i++)
00850   {
00851     temp_xy = getXYFromTurtleFrame(i);
00852
00853     sprites.turtle[i] = sprite_cut(sprites.turtle_uncut, temp_xy.x, temp_xy.y, TURTLE_SIDE,
      TURTLE_SIDE);
00854   }
00855
00856   // corazon
```

```
00857   path = make_sprite_path(SPRITE_HEART, path);
00858   sprites.heart = al_load_bitmap(path);
00859
00860   // fondo de menu
00861   path = make_sprite_path(SPRITE_MENU_HOME_BACK, path);
00862   sprites.menu[MENU_WINDOW_HOME].background = al_load_bitmap(path);
00863
00864   // botones con highlight, sin recortar
00865   path = make_sprite_path(SPRITE_MENU_HOME, path);
00866   sprites.menu[MENU_WINDOW_HOME].uncut = al_load_bitmap(path);
00867
00868   // se recortan los highlight
00869   for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00870   {
00871     sprites.menu[MENU_WINDOW_HOME].option[i] = sprite_cut(sprites.menu[MENU_WINDOW_HOME].uncut,
00872                                                           MENU_OPTION_TOPLEFT_X,
00873                                                           MENU_OPTION_TOPLEFT_Y + offset,
00874                                                           MENU_OPTION_W,
00875                                                           MENU_OPTION_H);
00876   }
00877
00878   path = make_sprite_path(SPRITE_MENU_DIFF_BACK, path);
00879   sprites.menu[MENU_WINDOW_DIFFICULTY].background = al_load_bitmap(path);
00880
00881   path = make_sprite_path(SPRITE_MENU_DIFF, path);
00882   sprites.menu[MENU_WINDOW_DIFFICULTY].uncut = al_load_bitmap(path);
00883
00884   for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00885   {
00886     sprites.menu[MENU_WINDOW_DIFFICULTY].option[i] =
      sprite_cut(sprites.menu[MENU_WINDOW_DIFFICULTY].uncut,
00887                                                             MENU_OPTION_TOPLEFT_X,
00888                                                             MENU_OPTION_TOPLEFT_Y + offset,
00889                                                             MENU_OPTION_W,
00890                                                             MENU_OPTION_H);
00891   }
00892
00893   path = make_sprite_path(SPRITE_MENU_PAUSE_BACK, path);
00894   sprites.menu[MENU_WINDOW_PAUSE].background = al_load_bitmap(path);
00895
00896   path = make_sprite_path(SPRITE_MENU_PAUSE, path);
00897   sprites.menu[MENU_WINDOW_PAUSE].uncut = al_load_bitmap(path);
00898
00899   for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00900   {
00901     sprites.menu[MENU_WINDOW_PAUSE].option[i] = sprite_cut(sprites.menu[MENU_WINDOW_PAUSE].uncut,
00902                                                           MENU_OPTION_TOPLEFT_X,
00903                                                           MENU_OPTION_TOPLEFT_Y + offset,
00904                                                           MENU_OPTION_W,
00905                                                           MENU_OPTION_H);
00906   }
00907
00908   path = make_sprite_path(SPRITE_MENU_GAME_OVER_BACK, path);
00909   sprites.menu[MENU_WINDOW_GAME_OVER].background = al_load_bitmap(path);
00910
00911   path = make_sprite_path(SPRITE_MENU_GAME_OVER, path);
00912   sprites.menu[MENU_WINDOW_GAME_OVER].uncut = al_load_bitmap(path);
00913
00914   for (i = 0, offset = 0; i < MENU_STATE_MAX; i++, offset += MENU_OPTION_DELTA_Y)
00915   {
00916     sprites.menu[MENU_WINDOW_GAME_OVER].option[i] =
      sprite_cut(sprites.menu[MENU_WINDOW_GAME_OVER].uncut,
00917                                                             MENU_OPTION_TOPLEFT_X,
00918                                                             MENU_OPTION_TOPLEFT_Y + offset,
00919                                                             MENU_OPTION_W,
00920                                                             MENU_OPTION_H);
00921   }
00922
00923   path = make_sprite_path(SPRITE_CREDITS, path);
00924   sprites.credits = al_load_bitmap(path);
00925
00926   path = make_sprite_path(SPRITE_NAME, path);
00927   sprites.name = al_load_bitmap(path);
00928
00929   path = make_sprite_path(SPRITE_ICON, path);
00930   sprites.icon = al_load_bitmap(path);
00931
00932   path = make_sprite_path(SPRITE_DEAD, path);
00933   sprites.dead = al_load_bitmap(path);
00934
00935   path = make_sprite_path(SPRITE_COIN, path);
00936   sprites.coin.uncut = al_load_bitmap(path);
00937   for (i = 0; i < SPRITE_COIN_FRAMES; i++)
00938   {
00939     pair_xy_t coord = getXYFromCoinFrame(i);
00940     sprites.coin.frame[i] = sprite_cut(sprites.coin.uncut, coord.x, coord.y, SPRITE_COIN_SIDE,
      SPRITE_COIN_SIDE);
```

```
00941   }
00942
00943   path = make_sprite_path(SPRITE_SPLASH, path);
00944   sprites.splash.uncut = al_load_bitmap(path);
00945   for (i = 0; i < SPRITE_SPLASH_FRAMES; i++)
00946   {
00947     pair_xy_t coord = getXYFromSplashFrame(i);
00948     sprites.splash.frame[i] = sprite_cut(sprites.splash.uncut, coord.x, coord.y, SPRITE_SPLASH_W,
      SPRITE_SPLASH_H);
00949   }
00950
00951   path = make_sprite_path(SPRITE_BORDER, path);
00952   sprites.border = al_load_bitmap(path);
00953
00954   free(path);
00955 }
00956
00957 static char *make_sprite_path(char *file_name, char *prev_str)
00958 {
00959   if (prev_str != NULL)
00960     free(prev_str);
00961
00962   char *path = NULL;
00963
00964   int str_size = sizeof(PATH_SPRITES) + strlen(file_name) + sizeof(EXTENSION_SPRITES) + 1;
00965
00966   path = malloc(str_size);
00967   must_init(path, file_name);
00968
00969   memset(path, 0, str_size);
00970
00971   strcat(path, PATH_SPRITES);
00972   strcat(path, file_name);
00973   strcat(path, EXTENSION_SPRITES);
00974
00975   return path;
00976 }
00977
00978 static void sprites_deinit(void)
00979 {
00980   int i, j;
00981
00982   al_destroy_bitmap(sprites.frog_uncut);
00983
00984   for (i = 0; i < FROG_FRAMES; i++)
00985     al_destroy_bitmap(sprites.frog[i]);
00986
00987   al_destroy_bitmap(sprites.background);
00988
00989   al_destroy_bitmap(sprites.log);
00990
00991   for (i = 0; i < CAR_TYPE_N; i++)
00992     al_destroy_bitmap(sprites.car[i]);
00993
00994   al_destroy_bitmap(sprites.turtle_uncut);
00995
00996   for (i = 0; i < TURTLE_FRAMES; i++)
00997     al_destroy_bitmap(sprites.turtle[i]);
00998
00999   al_destroy_bitmap(sprites.heart);
01000
01001   for (i = 0; i < MENU_WINDOW_MAX; i++)
01002   {
01003     al_destroy_bitmap(sprites.menu[i].background);
01004     al_destroy_bitmap(sprites.menu[i].uncut);
01005
01006     for (j = 0; j < MENU_STATE_MAX; j++)
01007     {
01008       al_destroy_bitmap(sprites.menu[i].option[j]);
01009     }
01010   }
01011
01012   al_destroy_bitmap(sprites.credits);
01013
01014   al_destroy_bitmap(sprites.name);
01015
01016   al_destroy_bitmap(sprites.icon);
01017
01018   al_destroy_bitmap(sprites.dead);
01019
01020   al_destroy_bitmap(sprites.coin.uncut);
01021   for (i = 0; i < SPRITE_COIN_FRAMES; i++)
01022     al_destroy_bitmap(sprites.coin.frame[i]);
01023
01024   al_destroy_bitmap(sprites.splash.uncut);
01025   for (i = 0; i < SPRITE_SPLASH_FRAMES; i++)
01026     al_destroy_bitmap(sprites.splash.frame[i]);
```

```
01027
01028   al_destroy_bitmap(sprites.border);
01029 }
01030
01031 static void audio_init(void)
01032 {
01033   // streams
01034   sounds.stream_state = SOUND_STREAM_STATE_NO_INIT;
01035
01036   // efectos de sonido
01037   must_init(init_sample(&sounds.samples.bonus, "bonus_alert"),
01038             "effect_bonus sample");
01039
01040   must_init(init_sample(&sounds.samples.click, "click"),
01041             "effect_click sample");
01042
01043   must_init(init_sample(&sounds.samples.crash, "crash"),
01044             "effect_crash sample");
01045
01046   must_init(init_sample(&sounds.samples.drowned, "fall_in_water"),
01047             "effect_drowned sample");
01048
01049   must_init(init_sample(&sounds.samples.goal, "goal_reached"),
01050             "effect_goal sample");
01051
01052   must_init(init_sample(&sounds.samples.jump, "jump_original"),
01053             "effect_jump sample");
01054
01055   must_init(init_sample(&sounds.samples.low_time, "low_time"),
01056             "effect_low_time sample");
01057
01058   must_init(init_sample(&sounds.samples.run_completed, "run_completed"),
01059             "effect_run_completed sample");
01060
01061   must_init(init_sample(&sounds.samples.menu_enter, "menu_enter"),
01062             "effect_menu_enter sample");
01063
01064   must_init(init_sample(&sounds.samples.new_max_score, "new_max_score"),
01065             "effect_new_max_score sample");
01066
01067   must_init(init_sample(&sounds.samples.exiting, "saliendo"),
01068             "effect_saliendo sample");
01069
01070   must_init(init_sample(&sounds.samples.no_time, "no_time"),
01071             "effect_no_time sample");
01072
01073   must_init(init_sample(&sounds.samples.coin_drop, "coin_drop"),
01074             "effect_coin_drop sample");
01075 }
01076
01077 static void audio_deinit(void)
01078 {
01079   if (sounds.stream_state != SOUND_STREAM_STATE_NO_INIT)
01080     al_destroy_audio_stream(sounds.stream);
01081
01082   al_destroy_sample(sounds.samples.bonus);
01083   al_destroy_sample(sounds.samples.click);
01084   al_destroy_sample(sounds.samples.crash);
01085   al_destroy_sample(sounds.samples.drowned);
01086   al_destroy_sample(sounds.samples.goal);
01087   al_destroy_sample(sounds.samples.jump);
01088   al_destroy_sample(sounds.samples.low_time);
01089   al_destroy_sample(sounds.samples.run_completed);
01090   al_destroy_sample(sounds.samples.menu_enter);
01091   al_destroy_sample(sounds.samples.no_time);
01092   al_destroy_sample(sounds.samples.coin_drop);
01093 }
01094
01095 static bool init_audio_stream(const char *file, float gain)
01096 {
01097   if (file == NULL)
01098     return false;
01099
01100   ALLEGRO_AUDIO_STREAM **pt = &sounds.stream;
01101   unsigned char *state = &sounds.stream_state;
01102
01103   int str_size = sizeof(PATH_SOUND_STREAMS) + strlen(file) + sizeof(EXTENSION_SOUND_STREAM) + 1;
01104   char *path = NULL;
01105
01106   // analisis de reproduccion y carga de stream "para que no explote todo"
01107   switch (*state)
01108   {
01109   case SOUND_STREAM_STATE_PLAY:
01110     // pausa
01111     al_set_audio_stream_playing(*pt, false);
01112
01113   case SOUND_STREAM_STATE_PAUSE:
```

```
01114     // desacople del mixer
01115     al_detach_audio_stream(*pt);
01116     // destruccion
01117     al_destroy_audio_stream(*pt);
01118
01119   case SOUND_STREAM_STATE_NO_INIT:
01120     // armado del path del archivo
01121     path = malloc(str_size);
01122     must_init(path, file);
01123     memset(path, 0, str_size);
01124     strcat(path, PATH_SOUND_STREAMS);
01125     strcat(path, file);
01126     strcat(path, EXTENSION_SOUND_STREAM);
01127
01128     // carga del stream
01129     *pt = al_load_audio_stream(path, 2, 2048);
01130     if (*pt == NULL)
01131       return false;
01132
01133     free(path);
01134
01135     // modo de reproduccion
01136     al_set_audio_stream_playmode(*pt, ALLEGRO_PLAYMODE_LOOP);
01137
01138     // ganancia
01139     al_set_audio_stream_gain(*pt, gain);
01140
01141     // pausa
01142     al_set_audio_stream_playing(*pt, false);
01143
01144     //"para que suene" (acople al mixer)
01145     al_attach_audio_stream_to_mixer(sounds.stream, al_get_default_mixer());
01146
01147     // actualiza el nombre del ultimo stream inicializado
01148     strcpy(last_init_stream, file);
01149
01150     *state = SOUND_STREAM_STATE_PAUSE;
01151     break;
01152
01153   default:
01154     break;
01155   }
01156
01157   return true;
01158 }
01159
01160 static bool init_sample(ALLEGRO_SAMPLE **sample, const char *file)
01161 {
01162   if (file == NULL)
01163     return false;
01164
01165   int str_size = sizeof(PATH_SOUND_SAMPLES) + strlen(file) + sizeof(EXTENSION_SOUND_SAMPLE) + 1;
01166   char *path = NULL;
01167
01168   path = malloc(str_size);
01169   must_init(path, file);
01170   memset(path, 0, str_size);
01171   strcat(path, PATH_SOUND_SAMPLES);
01172   strcat(path, file);
01173   strcat(path, EXTENSION_SOUND_SAMPLE);
01174
01175   *sample = al_load_sample(path);
01176
01177   free(path);
01178
01179   if (*sample == NULL)
01180     return false;
01181
01182   return true;
01183 }
01184
01185 static void rick_init(void)
01186 {
01187   rick = algif_load_animation("../res/gifs/rick.gif");
01188   allegro_set_rick_flag(false);
01189 }
01190
01191 static void rick_deinit()
01192 {
01193   algif_destroy_animation(rick);
01194 }
```

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

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

Include dependency graph for allegro_stuff.h:

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

Data Structures

- struct sprites_menu_t
- struct sprites_t

  Estructura principal de spritesheets.

Macros

- #define FPS 60

Enumerations

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

Functions

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

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

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

  Desinicializaciones de allegro.
- void allegro_reinit_display (void)

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

  Desinicializa el display.
- unsigned char allegro_get_last_key (void)

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

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

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

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

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

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

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

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

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

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

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

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

  Pone negro el display.
- void allegro_draw_background (void)

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

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

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

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

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

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

Selecciona musica de menu. Comienza pausada.

- void allegro_sound_set_stream_pause_menu (void)

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

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

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

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

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

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

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

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

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

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

  Reproduce efecto de bonus.
- void allegro_sound_play_effect_click (void)

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

  Reproduce efecto de choque.
- void allegro_sound_play_effect_drowned (void)

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

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

  Reproduce efecto de salto.
- void allegro_sound_play_effect_low_time (void)

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

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

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

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

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

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

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

  Dibuja un contorno rectangular.
- void allegro_rick_on (void)

- bool allegro_get_rick_flag (void)
- void allegro_set_rick_flag (bool state)
- void allegro_rick_off (void)
- void allegro_rick_draw (void)

Variables

- sprites_t sprites

4.26.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-10

Copyright

Definition in file allegro_stuff.h.

4.26.2 Macro Definition Documentation

4.26.2.1 FPS `#define FPS 60`

Definition at line 37 of file allegro_stuff.h.

4.26.3 Enumeration Type Documentation

4.26.3.1 KEY_STATES `enum KEY_STATES`

Definition at line 43 of file allegro_stuff.h.

4.26.4 Function Documentation

4.26.4.1 allegro_clear_display() `void allegro_clear_display (`

```
void )
```

Pone negro el display.

Definition at line 456 of file allegro_stuff.c.

4.26.4.2 allegro_deinit_display() `void allegro_deinit_display (`

```
void )
```

Desinicializa el display.

Definition at line 380 of file allegro_stuff.c.

4.26.4.3 allegro_deinits() `void allegro_deinits (`

```
void )
```

Desinicializaciones de allegro.

Definition at line 334 of file allegro_stuff.c.

4.26.4.4 allegro_draw_background() `void allegro_draw_background (`

```
void )
```

Dibuja la imagen de fondo.

Definition at line 461 of file allegro_stuff.c.

4.26.4.5 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 736 of file allegro_stuff.c.

4.26.4.6 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 466 of file allegro_stuff.c.

4.26.4.7 allegro_get_event_queue() `ALLEGRO_EVENT_QUEUE * allegro_get_event_queue ( void )`

Devuelve puntero a la cola de eventos.

Definition at line 476 of file allegro_stuff.c.

4.26.4.8 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 389 of file allegro_stuff.c.

4.26.4.9 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 406 of file allegro_stuff.c.

4.26.4.10 allegro_get_rick_flag() `bool allegro_get_rick_flag ( void )`

Definition at line 754 of file allegro_stuff.c.

4.26.4.11 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 421 of file allegro_stuff.c.

4.26.4.12 allegro_get_var_event() `ALLEGRO_EVENT allegro_get_var_event ( void )`

Devuelve el evento de allegro.

Returns

ALLEGRO_EVENT

Definition at line 416 of file allegro_stuff.c.

4.26.4.13 allegro_get_var_font() `ALLEGRO_FONT * allegro_get_var_font ( void )`

Devuelve la fuente de allegro.

Returns

ALLEGRO_FONT

Definition at line 441 of file allegro_stuff.c.

4.26.4.14 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 446 of file allegro_stuff.c.

4.26.4.15 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 451 of file allegro_stuff.c.

4.26.4.16 allegro_get_var_redraw() `bool allegro_get_var_redraw ( void )`

Devuelve flag de renderización.

Returns

true Renderiza

false No renderiza

Definition at line 426 of file allegro_stuff.c.

4.26.4.17 allegro_inits() `void allegro_inits ( void )`

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

Definition at line 283 of file allegro_stuff.c.

4.26.4.18 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 471 of file allegro_stuff.c.

4.26.4.19 allegro_reinit_display() void allegro_reinit_display (void)

Reinicializa el display de allegro.

Definition at line 347 of file allegro_stuff.c.

4.26.4.20 allegro_rick_draw() void allegro_rick_draw (void)

Definition at line 771 of file allegro_stuff.c.

4.26.4.21 allegro_rick_off() void allegro_rick_off (void)

Definition at line 764 of file allegro_stuff.c.

4.26.4.22 allegro_rick_on() void allegro_rick_on (void)

Definition at line 746 of file allegro_stuff.c.

4.26.4.23 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 394 of file allegro_stuff.c.

4.26.4.24 allegro_set_rick_flag() `void allegro_set_rick_flag ( bool state )`

Parameters

state	

Definition at line 759 of file allegro_stuff.c.

4.26.4.25 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 431 of file allegro_stuff.c.

4.26.4.26 allegro_set_var_event() `void allegro_set_var_event ( ALLEGRO_EVENT event )`

Carga un evento de allegro.

Parameters

event	

Definition at line 481 of file allegro_stuff.c.

4.26.4.27 allegro_set_var_redraw() `void allegro_set_var_redraw ( bool state )`

Setea flag de renderizacion.

Parameters

state	true or false

Definition at line 436 of file allegro_stuff.c.

4.26.4.28 allegro_sound_pause_stream() `void allegro_sound_pause_stream ( void )`

Pausa la musica actual, si hay alguna seleccionada.

Definition at line 629 of file allegro_stuff.c.

4.26.4.29 allegro_sound_play_effect_bonus() `void allegro_sound_play_effect_bonus ( void )`

Reproduce efecto de bonus.

Definition at line 667 of file allegro_stuff.c.

4.26.4.30 allegro_sound_play_effect_click() `void allegro_sound_play_effect_click ( void )`

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

Definition at line 672 of file allegro_stuff.c.

4.26.4.31 allegro_sound_play_effect_coin_drop() `void allegro_sound_play_effect_coin_drop ( void )`

Reproduce efecto `de moneda tirada`

Definition at line 727 of file allegro_stuff.c.

4.26.4.32 allegro_sound_play_effect_crash() `void allegro_sound_play_effect_crash ( void )`

Reproduce efecto de choque.

Definition at line 677 of file allegro_stuff.c.

4.26.4.33 allegro_sound_play_effect_drowned() `void allegro_sound_play_effect_drowned ( void )`

Reproduce efecto de ahogado (toco agua)

Definition at line 682 of file allegro_stuff.c.

4.26.4.34 allegro_sound_play_effect_exiting() `void allegro_sound_play_effect_exiting ( void )`

Reproduce efecto de 'saliendo'.

Definition at line 717 of file allegro_stuff.c.

4.26.4.35 allegro_sound_play_effect_goal() `void allegro_sound_play_effect_goal ( void )`

Reproduce efecto de 'llego a la meta'.

Definition at line 687 of file allegro_stuff.c.

4.26.4.36 allegro_sound_play_effect_jump() `void allegro_sound_play_effect_jump ( void )`

Reproduce efecto de salto.

Definition at line 692 of file allegro_stuff.c.

4.26.4.37 allegro_sound_play_effect_low_time() `void allegro_sound_play_effect_low_time ( void )`

Reproduce efecto de 'queda poco tiempo'.

Definition at line 697 of file allegro_stuff.c.

4.26.4.38 allegro_sound_play_effect_menu_enter() `void allegro_sound_play_effect_menu_enter ( void )`

Reproduce efecto de 'menu enter'.

Definition at line 707 of file allegro_stuff.c.

4.26.4.39 allegro_sound_play_effect_new_max_score() `void allegro_sound_play_effect_new_max_score (`

```
void )
```

Reproduce efecto de 'new_max_score'.

Definition at line 712 of file allegro_stuff.c.

4.26.4.40 allegro_sound_play_effect_no_time() `void allegro_sound_play_effect_no_time (`

```
void )
```

Reproduce efecto de 'sin tiempo'.

Definition at line 722 of file allegro_stuff.c.

4.26.4.41 allegro_sound_play_effect_run_completed() `void allegro_sound_play_effect_run_completed (`

```
void )
```

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

Definition at line 702 of file allegro_stuff.c.

4.26.4.42 allegro_sound_play_stream() `void allegro_sound_play_stream (`

```
void )
```

Reproduce la musica actual, si hay alguna seleccionada.

Definition at line 620 of file allegro_stuff.c.

4.26.4.43 allegro_sound_restart_stream() `void allegro_sound_restart_stream (`

```
void )
```

Reinicia la musica actual, si hay alguna seleccionada.

Definition at line 638 of file allegro_stuff.c.

4.26.4.44 allegro_sound_set_stream_credits() `void allegro_sound_set_stream_credits (`

```
void )
```

Selecciona musica de creditos. Comienza pausada.

Definition at line 489 of file allegro_stuff.c.

4.26.4.45 allegro_sound_set_stream_gain_down() `void allegro_sound_set_stream_gain_down ( void )`

Reduce en 0.1 la ganancia de stream.

Definition at line 655 of file allegro_stuff.c.

4.26.4.46 allegro_sound_set_stream_gain_up() `void allegro_sound_set_stream_gain_up ( void )`

Aumenta en 0.1 la ganancia de stream.

Definition at line 646 of file allegro_stuff.c.

4.26.4.47 allegro_sound_set_stream_game_over() `void allegro_sound_set_stream_game_over ( void )`

Selecciona musica de game over. Comienza pausada.

Definition at line 585 of file allegro_stuff.c.

4.26.4.48 allegro_sound_set_stream_main_menu() `void allegro_sound_set_stream_main_menu ( void )`

Selecciona musica de menu. Comienza pausada.

Definition at line 505 of file allegro_stuff.c.

4.26.4.49 allegro_sound_set_stream_pause_menu() `void allegro_sound_set_stream_pause_menu ( void )`

Selecciona musica de pausa. Comienza pausada.

Definition at line 521 of file allegro_stuff.c.

4.26.4.50 allegro_sound_set_stream_playing() `void allegro_sound_set_stream_playing ( void )`

Selecciona musica de jugando. Comienza pausada.

Definition at line 553 of file allegro_stuff.c.

4.26.4.51 allegro_sound_set_stream_ranking() `void allegro_sound_set_stream_ranking (`

```
void )
```

Selecciona musica de ranking. Comienza pausada.

Definition at line 537 of file allegro_stuff.c.

4.26.4.52 allegro_sound_set_stream_rick() `void allegro_sound_set_stream_rick (`

```
void )
```

Definition at line 569 of file allegro_stuff.c.

4.26.4.53 allegro_sound_toggle_stream() `void allegro_sound_toggle_stream (`

```
void )
```

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

Definition at line 604 of file allegro_stuff.c.

4.26.4.54 allegro_wait_for_event() `ALLEGRO_EVENT_TYPE allegro_wait_for_event (`

```
void )
```

Espera a que ocurra un evento y lo devuelve.

Returns

ALLEGRO_EVENT_TYPE

Definition at line 399 of file allegro_stuff.c.

4.26.4.55 must_init() `void must_init (`

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

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

Parameters

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

Definition at line 274 of file allegro_stuff.c.

4.26.5 Variable Documentation

4.26.5.1 sprites `sprites_t sprites [extern]`

Definition at line 150 of file allegro_stuff.c.

4.27 allegro_stuff.h

Go to the documentation of this file.

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

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

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

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

```
#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.

4.28.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-22

Copyright

Definition in file display.c.

4.28.2 Macro Definition Documentation

4.28.2.1 CREDITS_SCROLL_SPEED `#define CREDITS_SCROLL_SPEED 1`

Definition at line 28 of file display.c.

4.28.2.2 RANKING_PLAYER_X `#define RANKING_PLAYER_X 90`

Definition at line 30 of file display.c.

4.28.2.3 RANKING_SCORE_X `#define RANKING_SCORE_X 500`

Definition at line 31 of file display.c.

4.28.2.4 RANKING_START_Y `#define RANKING_START_Y 100`

Definition at line 32 of file display.c.

4.28.3 Function Documentation

4.28.3.1 actualizarDisplay() `void actualizarDisplay ( )`

Actualiza el display de la plataforma.

Definition at line 58 of file display.c.

4.28.3.2 cargarCreditos() `void cargarCreditos ( void )`

Inicializa los cretidos en la plataforma.

Definition at line 140 of file display.c.

4.28.3.3 cargarRanking() `void cargarRanking ( void )`

Inicia muestreo de ranking en la plataforma.

Parameters

txt	

Definition at line 78 of file display.c.

4.28.3.4 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 73 of file display.c.

4.28.3.5 iniciarDisplay() `bool iniciarDisplay ( )`

Inicializa el display de la plataforma.

Returns

true Exito

false Error

Definition at line 48 of file display.c.

4.28.3.6 limpiarDisplay() `void limpiarDisplay ( )`

Limpia el display de la plataforma.

Definition at line 63 of file display.c.

4.28.3.7 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 145 of file display.c.

4.28.3.8 mostrarRanking() `void mostrarRanking ( void )`

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

Definition at line 135 of file display.c.

4.28.3.9 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 68 of file display.c.

4.28.3.10 reconfigurarDisplayOFF() `void reconfigurarDisplayOFF ( void )`

Reconfigura el display de la plataforma y lo deshabilita.

Definition at line 167 of file display.c.

4.28.3.11 reconfigurarDisplayON() `void reconfigurarDisplayON ( void )`

Reconfigura el display de la plataforma y lo habilita.

Definition at line 162 of file display.c.

4.29 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
00063 void limpiarDisplay()
00064 {
00065
00066 }
00067
00068 void mostrarTexto(char *txt, int pos)
00069 {
00070
00071 }
00072
00073 void dejarTexto(char *txt, int pos, bool repetir)
00074 {
00075
00076 }
00077
00078 void cargarRanking(void)
00079 {
00080   int i = 0;
00081
00082   int lines = getRankingLineas();
00083   char **names = getRankingNombres();
00084   unsigned long long *scores = getRankingPuntos();
00085
00086     allegro_clear_display();
00087
00088   al_draw_text( allegro_get_var_font(),
00089                               al_map_rgb(10,180,10),
00090                                   RANKING_PLAYER_X,
00091                                   60,
00092                                   0,
00093                                   "Jugador");
00094
00095     al_draw_text(   allegro_get_var_font(),
00096                               al_map_rgb(10,180,10),
00097                                   RANKING_SCORE_X,
00098                                   60,
00099                                   0,
00100                                   "Puntaje");
00101
00102     if(lines)
00103     {
00104         for (i = 0; i < lines; i++)
00105         {
00106             al_draw_textf(allegro_get_var_font(),
00107                               al_map_rgb(255, 255, 255),
00108                               RANKING_PLAYER_X,
00109                               RANKING_START_Y + i * 20,
00110                               0,
00111                               "%s", names[i]);
00112
00113             al_draw_textf(allegro_get_var_font(),
00114                               al_map_rgb(255, 255, 255),
00115                               RANKING_SCORE_X,
00116                               RANKING_START_Y + i * 20,
00117                               0,
00118                               "%lld", scores[i]);
00119         }
00120     }
00121
00122     else
00123     {
00124         al_draw_text(   allegro_get_var_font(),
00125                                     al_map_rgb(255,255,255),
00126                                     DISPLAY_W/2 - al_get_text_width(allegro_get_var_font(), "Ningún
      jugador registrado")/2,
00127                                     RANKING_START_Y,
00128                                     0,
00129                                     "Ningún jugador registrado");
00130     }
00131
00132   al_flip_display();
00133 }
```

```
00134
00135 void mostrarRanking(void)
00136 {
00137
00138 }
00139
00140 void cargarCreditos(void)
00141 {
00142   credits_scroll_cont = 0;
00143 }
00144
00145 void mostrarCreditos(void)
00146 {
00147   if (allegro_get_var_redraw())
00148   {
00149
00150     credits_scroll_cont -= CREDITS_SCROLL_SPEED;
00151     if (credits_scroll_cont == -CREDITS_SCREEN_FINAL)
00152       credits_scroll_cont = CREDITS_SCREEN_START;
00153
00154     allegro_clear_display();
00155     al_draw_bitmap(sprites.credits, 0, credits_scroll_cont, 0);
00156     al_flip_display();
00157
00158     allegro_set_var_redraw(false);
00159   }
00160 }
00161
00162 void reconfigurarDisplayON(void)
00163 {
00164   allegro_reinit_display();
00165 }
00166
00167 void reconfigurarDisplayOFF(void)
00168 {
00169   allegro_deinit_display();
00170 }
```

4.30 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 #include <stdio.h>
00023
00024 /*********************************************************************************
00025  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00026  *********************************************************************************/
00027
00028 #define CASTEAR_POSICION(pos) ((pos) == POS_CREDITOS ? POS_MSJ3 : (((pos) == POS_OPCION) || ((pos) ==
     POS_RANKING_2) ? POS_MSJ2 : POS_MSJ1))
00029
00030 /*********************************************************************************
00031  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00032  *********************************************************************************/
00033
00034 /*********************************************************************************
00035  * VARIABLES WITH GLOBAL SCOPE
00036  *********************************************************************************/
00037
00038 matriz_t disp_matriz;
00039
00040 /*********************************************************************************
00041  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00042  *********************************************************************************/
00043
00050 static void *threadTextoDisplay(void *ptr);
00051 static void *threadPresentacion(void *ptr);
00052 static void ulltoa(unsigned long long num, char *str);
00053
00054 /*********************************************************************************
00055  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
```

```
00056  ******************************************************************************/
00057
00058 static const clock_t TIEMPO_SLEEP_DISPLAY = CLOCKS_PER_SEC » 2;
00059
00060 /*******************************************************************************
00061  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00062  ******************************************************************************/
00063
00064 static pthread_mutex_t lock;
00065 static pthread_t ttextodisplay, tpresentacion;
00066 static mensaje_t texto1, texto2, texto3;
00067 static bool thread_encendido, thread_presentacion_encendido;
00068
00069 static int lines, i;
00070 static char **names;
00071 static unsigned long long *scores;
00072
00073 static char *creditos_cadenas[] = {"PROGRAMACION TPF 2021 1C", "FROGGER", "AUTORES", "ALEJANDRO HEIR",
       "FRANCO AGGRIPINO", "MATIAS ALVAREZ", "TOMAS CASTRO"};
00074
00075 /*******************************************************************************
00076  *******************************************************************************
00077                         GLOBAL FUNCTION DEFINITIONS
00078  *******************************************************************************
00079  ******************************************************************************/
00080
00081 bool iniciarDisplay()
00082 {
00083   if (pthread_mutex_init(&lock, NULL) != 0)
00084   {
00085     return 1;
00086   }
00087
00088   disp_init();  // inicializa el display
00089   disp_clear(); // limpia todo el display
00090
00091   return 0;
00092 }
00093
00094 void actualizarDisplay()
00095 {
00096   pthread_mutex_lock(&lock);
00097   for (int i = DISP_MIN; i <= (DISP_MAX_Y); i++)
00098     for (int j = DISP_MIN; j <= (DISP_MAX_X); j++)
00099       disp_write((dcoord_t){j, i}, disp_matriz[i] & (0x8000 » j));
00100
00101   disp_update();
00102   pthread_mutex_unlock(&lock);
00103 }
00104
00105 void limpiarDisplay()
00106 {
00107   texto1.habilitacion = false;
00108   texto2.habilitacion = false;
00109   texto3.habilitacion = false;
00110
00111   if (thread_encendido)
00112   {
00113     thread_encendido = false;
00114     pthread_join(ttextodisplay, NULL);
00115   }
00116   if (thread_presentacion_encendido)
00117   {
00118     thread_presentacion_encendido = false;
00119     pthread_join(tpresentacion, NULL);
00120   }
00121
00122   limpiarMatriz(disp_matriz);
00123   actualizarDisplay();
00124 }
00125
00126 void mostrarTexto(char *txt, int pos)
00127 {
00128   int posicion = CASTEAR_POSICION(pos);
00129   mensaje_t msj = mensaje(txt, posicion, false);
00130   while (!renglonIzquierdoLibre(&msj))
00131   {
00132     usleep(TIEMPO_SLEEP_DISPLAY);
00133     moverMensaje(&msj);
00134     copiarMatrizRenglon(disp_matriz, msj.renglon, msj.posicion);
00135     actualizarDisplay();
00136   }
00137 }
00138
00139 void dejarTexto(char *txt, int pos, bool repetir)
00140 {
00141   int posicion = CASTEAR_POSICION(pos);
```

```
00142
00143   switch (posicion)
00144   {
00145   case POS_MSJ1:
00146     texto1 = mensaje(txt, posicion, repetir);
00147     break;
00148   case POS_MSJ2:
00149     texto2 = mensaje(txt, posicion, repetir);
00150     break;
00151   default:
00152     limpiarDisplay();
00153     texto3 = mensaje(txt, posicion, repetir);
00154   }
00155
00156   if (!thread_encendido)
00157   {
00158     thread_encendido = true;
00159     pthread_create(&ttextodisplay, NULL, threadTextoDisplay, NULL);
00160   }
00161 }
00162
00163 void cargarRanking(void)
00164 {
00165   borrarRenglon(texto2.renglon);
00166   lines = getRankingLineas();
00167   if (lines <= 0)
00168     dejarTexto("NINGUNA PARTIDA COMPLETADA AUN", POS_MSJ2, true);
00169   else
00170   {
00171     names = getRankingNombres();
00172     scores = getRankingPuntos();
00173     i = 0;
00174   }
00175 }
00176
00177 void mostrarRanking(void)
00178 {
00179   if (renglonIzquierdoLibre(&texto2) && lines > 0)
00180   { // si se acabó lo que tenía para mostrar abajo, busco la siguiente posición
00181     renglon_t r = {0};
00182     uintARenglon(i + 1, r);
00183     copiarMatrizRenglon(disp_matriz, r, POS_MSJ1); // se pone la posición en el renglón de arriba
00184     actualizarDisplay();
00185
00186     char score_str[L_MAX], puesto_msj[L_MAX];
00187
00188     strcpy(puesto_msj, names[i]);
00189     strcat(puesto_msj, " ");
00190     ulltoa(scores[i], score_str);
00191     strcat(puesto_msj, score_str);           // se arma un string con el nombre de jugador y la
    puntuación
00192     dejarTexto(puesto_msj, POS_MSJ2, false); // se muestra el string en la posición de abajo hasta que
00193
00194     if (++i >= lines) // apunto a la siguiente posición
00195       i = 0;
00196   }
00197 }
00198
00199 void cargarCreditos()
00200 {
00201   i = 0;
00202 }
00203
00204 void mostrarCreditos(void)
00205 {
00206   if ((renglonIzquierdoLibre(&texto3) && !thread_presentacion_encendido) && i < 7)
00207   {
00208     switch (i)
00209     {
00210     case 1:
00211       limpiarDisplay();
00212       thread_presentacion_encendido = true;
00213       pthread_create(&tpresentacion, NULL, threadPresentacion, NULL);
00214       break;
00215     case 2:
00216       pthread_join(tpresentacion, NULL);
00217     default:
00218       dejarTexto(creditos_cadenas[i], POS_CREDITOS, false);
00219     }
00220     i++;
00221   }
00222 }
00223
00224 void reconfigurarDisplayON(void)
00225 {
00226 }
00227
```

```
00228 void reconfigurarDisplayOFF(void)
00229 {
00230 }
00231
00232 /********************************************************************************
00233  ********************************************************************************
00234                         LOCAL FUNCTION DEFINITIONS
00235  ********************************************************************************
00236  ********************************************************************************/
00237
00238 static void *threadTextoDisplay(void *ptr)
00239 {
00240   while (thread_encendido && (texto1.habilitacion || texto2.habilitacion || texto3.habilitacion))
00241   {
00242     usleep(TIEMPO_SLEEP_DISPLAY);
00243     if (texto1.habilitacion)
00244     {
00245       moverMensaje(&texto1);
00246       copiarMatrizRenglon(disp_matriz, texto1.renglon, POS_MSJ1);
00247     }
00248     if (texto2.habilitacion)
00249     {
00250       moverMensaje(&texto2);
00251       copiarMatrizRenglon(disp_matriz, texto2.renglon, POS_MSJ2);
00252     }
00253     if (texto3.habilitacion)
00254     {
00255       moverMensaje(&texto3);
00256       copiarMatrizRenglon(disp_matriz, texto3.renglon, POS_MSJ3);
00257     }
00258     actualizarDisplay();
00259   }
00260
00261   thread_encendido = false;
00262   pthread_exit(NULL);
00263 }
00264
00265 static void *threadPresentacion(void *ptr)
00266 {
00267   int coordenadas[][2] = {{0, 0}, {4, 1}, {8, 3}, {1, 8}, {5, 9}, {9, 10}, {13, 11}};
00268   int j;
00269   for (j = 0; j < 14 && thread_presentacion_encendido; j++)
00270   {
00271     matriz_t letra_matriz;
00272     charAMatriz(creditos_cadenas[1][j % 7], letra_matriz, coordenadas[j % 7]);
00273     if (j >= 7)
00274       matrizXor(disp_matriz, letra_matriz);
00275     else
00276       matrizOr(disp_matriz, letra_matriz);
00277     actualizarDisplay();
00278     usleep(TIEMPO_SLEEP_DISPLAY);
00279   }
00280
00281   thread_presentacion_encendido = false;
00282
00283   return NULL;
00284 }
00285
00286 static void ulltoa(unsigned long long num, char *str)
00287 {
00288   unsigned long long sum = num;
00289   int i = 0;
00290   int digit;
00291   do
00292   {
00293     digit = sum % 10;
00294     str[i++] = '0' + digit;
00295     sum /= 10;
00296   } while (sum);
00297   str[i--] = '\0';
00298
00299   int j = 0;
00300   char ch;
00301   while (i > j)
00302   {
00303     ch = str[i];
00304     str[i--] = str[j];
00305     str[j++] = ch;
00306   }
00307 }
```

4.31 entities.c

```
00001 /********************************************************************************
```

```
00002  * INCLUDE HEADER FILES
00003  *****************************************************************************/
00004
00005 #include "entities.h"
00006 #include "allegro_stuff.h"
00007 #include "geometry.h"
00008 #include "game_data.h"
00009
00010 /*****************************************************************************
00011  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00012  *****************************************************************************/
00013
00014 //#define DEBUG_ENTITIES_TEXT
00015
00016 #define LOGS_SPAWN_MIN 1
00017 #define LOGS_SPAWN_MAX 3
00018 #define LOGS_SPAWN_FRAMES 60
00019 #define LOGS_BASE_SPEED 1
00020 #define LOGS_MAX_USED 7
00021 #define LOGS_EXTRA_SEPARATOR LOG_W / 2
00022
00023 #define CARS_SPAWN_MIN 2
00024 #define CARS_SPAWN_FRAMES 60
00025 #define CARS_BASE_SPEED 1
00026 #define CARS_MAX_USED 15
00027 #define CAR_SPEED_INCREASE 2
00028 #define CAR_WAIT_INCREASE 1
00029 #define CARS_EXTRA_SEPARATOR CAR_W * 2
00030
00031 #define TURTLES_MIN_PER_PACK 1
00032 #define TURTLES_MAX_PER_PACK 2
00033 #define TURTLES_SPAWN_FRAMES 60 // cada cuantos frames spawnean
00034 #define TURTLES_SPAWN_MIN 1     // minimas a spawnear de una
00035 #define TURTLES_SPAWN_MAX 2     // maximas a spawnear de una
00036 #define TURTLES_MAX_USED 7      // maximas en pantalla
00037 #define TURTLES_BASE_SPEED 2
00038 #define TURTLES_FRAME_TIMEOUT_SURFACE 10    // cuanto "tiempo" dura un frame dibujado antes de pasar
      al siguiente
00039 #define TURTLES_FRAME_TIMEOUT_GOING_DOWN 50 // tiempo por frame al sumergirse
00040 #define TURTLES_FRAME_TIMEOUT_WATER 20      // tiempo por frame para mostrarse bajo el agua
00041 #define TURTLES_FRAME_TIMEOUT_GOING_UP 10   // tiempo por frame para mostrarse saliendo del agua
00042 #define TURTLES_SURFACE_FRAMES_MIN 80       // minimo "tiempo" en superficie
00043 #define TURTLES_SURFACE_FRAMES_MAX 500      // maximo "tiempo" en superficie
00044 #define TURTLES_WATER_FRAMES_MIN 60         // minimo "tiempo" bajo el agua
00045 #define TURTLES_WATER_FRAMES_MAX 100        // maximo "tiempo" bajo el agua
00046 #define TURTLES_EXTRA_SEPARATOR TURTLE_SIDE * 2
00047
00048 #define COIN_SPAWN_FRAMES_MIN 300   // mínimo tiempo para respawnear coin
00049 #define COIN_SPAWN_FRAMES_MAX 600   // maximo tiempo para respawnear coin
00050 #define COIN_DESPAWN_FRAMES_MIN 700 // mínimo tiempo para sacar coin
00051 #define COIN_DESPAWN_FRAMES_MAX 900 // maximo tiempo para sacar coin
00052 #define COIN_FRAME_RATE 10          // cada cuanto gira la coin
00053 #define COIN_FRAMES_TO_WARN_A 250   // frames previos al despawneo cuando empieza a titilar
00054 #define COIN_FRAMES_TO_WARN_B 100
00055 #define COIN_WARNING_FRAMES_A 20 // blink rate
00056 #define COIN_WARNING_FRAMES_B 10
00057
00058 #define SPRITE_DEAD_TIMEOUT 80 // frames que permanece en pantalla el sprite de muerte
00059
00060 #define SPRITE_SPLASH_RATE 10 // cada cuanto avanza un frame la animacion
00061
00062 /*****************************************************************************
00063  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00064  *****************************************************************************/
00065
00066 typedef struct
00067 {
00068   int x;
00069   int y;
00070   int moving;
00071   int facing;
00072   int steps;
00073   unsigned char state;
00074   unsigned char next_action;
00075
00076 } frog_t;
00077
00078 typedef struct
00079 {
00080   int x; // Posicion del auto
00081   int y;
00082   int lane;      // Carril del auto.
00083   int dx;        // Velocidad del auto.
00084   CAR_TYPE type; // Tipo de auto.
00085   int length;    // Largo del auto.
00086   int count;
00087   bool fast;
```

```
00088   bool used; // Marca disponibilidad en el array.
00089 } car_t;
00090
00091 typedef struct
00092 {
00093   int x;
00094   int y;
00095   int lane;
00096   int dx;
00097   bool used;
00098
00099 } log_t;
00100
00101 typedef struct
00102 {
00103   int x;                          // coordenada x
00104   int y;                          // coordenada y
00105   int lane;                       // carril
00106   int dx;                         // velocidad
00107   bool used;                      // flag de usada o no
00108   unsigned char turtles_in_pack; // cantidad de tortugas en el paquete
00109
00110   struct
00111   {
00112     unsigned char frame;  // contador que indica en qué frame de la animación se está (de 1 a
       TURTLES_FRAMES)
00113     unsigned int timeout; // timeout interno para cambiar de frame
00114     unsigned int cont;    // contador interno de frames de juego ejecutados
00115   } fx;
00116
00117   int wide;            // ancho del paquete, proporcional a turtles_in_pack y a TURTLES_SIDE
00118   unsigned char state; // estado (enum TURTLE_STATES)
00119 } turtle_pack_t;
00120
00121 typedef struct
00122 {
00123   int x;
00124   int y;
00125   bool used; // flag de usada o no
00126   struct
00127   {
00128     unsigned int frame_cont;  // contador de frame a mostrar
00129     unsigned int timeout;     // Para spawneo y despawneo
00130     unsigned int blink_timer; // Para titilar coin antes de sacarla
00131     unsigned int cont;        // contador interno de frames de juego ejecutados
00132     bool flag;                // Para indicar si debe parpadear o no
00133   } fx;
00134
00135 } coin_t;
00136
00137 enum TURTLE_STATES
00138 {
00139   TURTLE_STATE_SURFACE,
00140   TURTLE_STATE_GOING_DOWN,
00141   TURTLE_STATE_WATER,
00142   TURTLE_STATE_GOING_UP
00143 };
00144
00145 enum FROG_STATES
00146 {
00147   FROG_STATE_ROAD,
00148   FROG_STATE_WATER,
00149   FROG_STATE_LOG,
00150   FROG_STATE_TURTLE,
00151   FROG_STATE_GOAL,
00152   FROG_STATE_GOAL_COIN, // meta con coin
00153   FROG_STATE_CRASH_CAR,
00154   FROG_STATE_CRASH_WALL,
00155   FROG_STATE_BOUNCING_WALL // rebota contra algun borde
00156 };
00157
00158 // Estructur para administrar el sprite de muerte
00159 static struct
00160 {
00161   bool flag;          // para indicar graficar
00162   unsigned int timer; // contador para permanecer en pantalla
00163   unsigned int x;
00164   unsigned int y;
00165 } corpse_fx;
00166
00167 // Estructura para administrar el efecto de caida en agua
00168 static struct
00169 {
00170   bool flag;               // usado o no
00171   unsigned int frame_cont; // contador de frame a mostrar
00172   unsigned int cont;       // contador de ejecucion
00173   unsigned int x;          // X topleft
```

```
00174   unsigned int y;          // Y topleft
00175
00176 } splash_fx;
00177
00178 /*********************************************************************************
00179  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00180  *********************************************************************************/
00181
00186 static void frog_init(void);
00187
00192 static void frog_update(void);
00193
00198 static void frog_draw(void);
00199
00204 static void logs_init(void);
00205
00210 static void logs_update(void);
00211
00216 static void logs_draw(void);
00217
00222 static void cars_init(void);
00223
00228 static void cars_update(void);
00229
00234 static void cars_draw(void);
00235
00240 static void turtles_init(void);
00241
00246 static void turtles_update(void);
00247
00252 static void turtles_draw(void);
00253
00258 static void coin_init(void);
00259
00264 static void coin_update(void);
00265
00270 static void coin_draw(void);
00271
00276 // static void fix_frog_pos(void);
00277
00282 static void fix_frog_coord_y(void);
00283
00292 static bool is_frog_in_goal(void);
00293
00300 static void corpse_init(int x, int y);
00301
00306 static void corpse_update(void);
00307
00312 static void corpse_draw(void);
00313
00320 static void splash_init(int x, int y);
00321
00326 static void splash_update(void);
00327
00332 static void splash_draw(void);
00333
00334 /*********************************************************************************
00335  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00336  *********************************************************************************/
00337
00338 // Rana
00339 static frog_t frog;
00340
00341 // Array de troncos
00342 static log_t log[LOGS_MAX_USED];
00343
00344 // Array de autos
00345 static car_t car[CARS_MAX_USED];
00346
00347 // Array de paquetes de tortugas
00348 static turtle_pack_t turtle_pack[TURTLES_MAX_USED];
00349
00350 // Coin
00351 static coin_t coin;
00352
00353 // Contador de frames ejecutados
00354 static unsigned long game_frames;
00355
00356 // Carriles variables.
00357 static unsigned char normal_diff_lane;
00358 static unsigned char hard_diff_lane_1;
00359 static unsigned char hard_diff_lane_2;
00360
00361 // Maximo de autos spawneados.
00362
00363 static unsigned char cars_spawn_max;
00364
```

```
00365 /*****************************************************************************
00366  *****************************************************************************
00367             GLOBAL FUNCTION DEFINITIONS
00368  *****************************************************************************
00369  *****************************************************************************/
00370
00371 void entities_init(void)
00372 {
00373   frog_init();
00374   logs_init();
00375   cars_init();
00376   turtles_init();
00377   coin_init();
00378
00379   game_frames = 0;
00380
00381   corpse_fx.flag = false;
00382
00383   splash_fx.flag = false;
00384 }
00385
00386 void entities_update()
00387 {
00388   game_frames = game_data_get_frames();
00389
00390   frog_update();
00391   logs_update();
00392   cars_update();
00393   turtles_update();
00394   coin_update();
00395
00396   corpse_update();
00397   splash_update();
00398 }
00399
00400 void entities_draw()
00401 {
00402   logs_draw();
00403   cars_draw();
00404   turtles_draw();
00405   coin_draw();
00406
00407   splash_draw();
00408   corpse_draw();
00409
00410   //"frog siempre a lo ultimo, para que se vea"
00411   frog_draw();
00412 }
00413
00414 void entities_move_frog(unsigned char direction)
00415 {
00416   if (direction == DIRECTION_DOWN || direction == DIRECTION_LEFT ||
00417       direction == DIRECTION_UP || direction == DIRECTION_RIGHT)
00418   {
00419     frog.next_action = direction;
00420   }
00421 }
00422
00423 /*****************************************************************************
00424  *****************************************************************************
00425             LOCAL FUNCTION DEFINITIONS
00426  *****************************************************************************
00427  *****************************************************************************/
00428
00429 static void frog_init(void)
00430 {
00431   frog.x = CELL_START_FROG_X;
00432   frog.y = CELL_START_FROG_Y;
00433   frog.moving = false;
00434   frog.facing = DIRECTION_UP;
00435   frog.steps = 0;
00436   frog.state = FROG_STATE_ROAD;
00437   frog.next_action = DIRECTION_NONE;
00438 }
00439
00440 static void frog_update(void)
00441 {
00442   int i;
00443
00444   bool interaction_flag = false;
00445
00446   if (!frog.moving)
00447   {
00448     if (frog.next_action == DIRECTION_DOWN || frog.next_action == DIRECTION_LEFT ||
00449         frog.next_action == DIRECTION_UP || frog.next_action == DIRECTION_RIGHT)
00450     {
00451       frog.facing = frog.next_action;
```

```
00452       frog.moving = true;
00453       frog.next_action = DIRECTION_NONE;
00454       allegro_sound_play_effect_jump();
00455     }
00456   }
00457
00458   else if (frog.moving)
00459   {
00460
00461     if (frog.facing == DIRECTION_LEFT)
00462       frog.x -= STEP_FRACTION_SIZE;
00463     else if (frog.facing == DIRECTION_RIGHT)
00464       frog.x += STEP_FRACTION_SIZE;
00465     else if (frog.facing == DIRECTION_UP)
00466       frog.y -= STEP_FRACTION_SIZE;
00467     else if (frog.facing == DIRECTION_DOWN)
00468       frog.y += STEP_FRACTION_SIZE;
00469
00470     if (++frog.steps >= STEP_RATIO)
00471     {
00472       frog.steps = 0;
00473       frog.moving = false;
00474
00475       fix_frog_coord_y();
00476     }
00477   }
00478
00479   // donde esta parada
00480   if (!frog.moving)
00481   {
00482     unsigned int y_no_offset = frog.y - FROG_OFFSET_Y;
00483
00484     // en alguna fila de descanso o de autos
00485     if (y_no_offset >= CELL_H * (lanes_cars[0] - 1) && y_no_offset <= DISPLAY_H - CELL_H)
00486       frog.state = FROG_STATE_ROAD;
00487
00488     // en alguna fila de agua. Luego se actualiza si es sobre tronco o turtle
00489     else if (y_no_offset >= CELL_H * 2 && y_no_offset <= CELL_H * (lanes_cars[0] - 1))
00490       frog.state = FROG_STATE_WATER;
00491
00492     // choque contra alguno de los muros superiores, o llegada bien a un goal
00493     else if (y_no_offset < CELL_H * 2)
00494     {
00495       if (!is_frog_in_goal())
00496       {
00497         frog.state = FROG_STATE_CRASH_WALL;
00498       }
00499
00500       else
00501       {
00502         frog.state = FROG_STATE_GOAL;
00503
00504         // colision con coin
00505         if (coin.used)
00506         {
00507           if (collideShort(coin.x,
00508                            coin.y,
00509                            SPRITE_COIN_SIDE,
00510                            SPRITE_COIN_SIDE,
00511                            frog.x,
00512                            frog.y,
00513                            FROG_W,
00514                            FROG_H))
00515           {
00516             frog.state = FROG_STATE_GOAL_COIN;
00517             coin.used = false;
00518           }
00519         }
00520       }
00521
00522       interaction_flag = true;
00523     }
00524
00525     /*---*/
00526
00527     if (!interaction_flag)
00528     {
00529       // colision con autos
00530       for (i = 0; i < CARS_MAX_USED; i++)
00531       {
00532         if (!car[i].used)
00533           continue;
00534
00535         if (collideShort(car[i].x,
00536                          car[i].y,
00537                          car[i].length,
00538                          CAR_H,
```

```
00539                           frog.x,
00540                           frog.y,
00541                           FROG_W,
00542                           FROG_H))
00543         {
00544           frog.state = FROG_STATE_CRASH_CAR;
00545           interaction_flag = true;
00546           break; // no puede chocar con 2 autos a la vez
00547         }
00548       }
00549     }
00550   }
00551
00552   if (!interaction_flag)
00553   {
00554     // esta en algun tronco?
00555     for (i = 0; i < LOGS_MAX_USED; i++)
00556     {
00557       if (!log[i].used)
00558         continue;
00559
00560       if (insideShortScaled(log[i].x,
00561                             log[i].y,
00562                             LOG_W,
00563                             LOG_H,
00564                             frog.x,
00565                             frog.y,
00566                             FROG_W,
00567                             FROG_H,
00568                             INSERTION_FACTOR))
00569       {
00570         frog.x += log[i].dx;
00571         frog.state = FROG_STATE_LOG;
00572         interaction_flag = true;
00573         break; // no puede estar en 2 troncos a la vez
00574       }
00575     }
00576   }
00577
00578   if (!interaction_flag)
00579   {
00580     // esta en algun turtle_pack?
00581     for (i = 0; i < TURTLES_MAX_USED; i++)
00582     {
00583       // Omite si el pack no esta usado o si esta bajo agua
00584       if (!turtle_pack[i].used || turtle_pack[i].state == TURTLE_STATE_WATER)
00585         continue;
00586
00587       if (insideShortScaled(turtle_pack[i].x,
00588                             turtle_pack[i].y,
00589                             turtle_pack[i].wide,
00590                             TURTLE_SIDE,
00591                             frog.x,
00592                             frog.y,
00593                             FROG_W,
00594                             FROG_H,
00595                             INSERTION_FACTOR))
00596       {
00597         frog.x += turtle_pack[i].dx;
00598         frog.state = FROG_STATE_TURTLE;
00599         interaction_flag = true;
00600         break; // no puede estar en 2 packs a la vez
00601       }
00602     }
00603   }
00604
00605   // revision de limites
00606   if (frog.x < FROG_MIN_X)
00607     frog.x = FROG_MIN_X;
00608   else if (frog.x > FROG_MAX_X)
00609     frog.x = FROG_MAX_X;
00610   else if (frog.y < FROG_MIN_Y)
00611     frog.y = FROG_MIN_Y;
00612   else if (frog.y > FROG_MAX_Y)
00613     frog.y = FROG_MAX_Y;
00614
00615   switch (frog.state)
00616   {
00617   case FROG_STATE_WATER:
00618     game_data_subtract_live();
00619     allegro_sound_play_effect_drowned();
00620
00621     splash_init(frog.x, frog.y);
00622
00623     frog_init();
00624
00625     break;
```

```
00626
00627   case FROG_STATE_CRASH_CAR:
00628     game_data_subtract_live();
00629     allegro_sound_play_effect_crash();
00630
00631     corpse_init(frog.x, frog.y);
00632
00633     frog_init();
00634
00635     break;
00636
00637   case FROG_STATE_CRASH_WALL:
00638     game_data_subtract_live();
00639     allegro_sound_play_effect_crash();
00640
00641     corpse_init(frog.x, frog.y);
00642
00643     frog_init();
00644
00645     break;
00646
00647   case FROG_STATE_GOAL:
00648     game_data_add_run_time_goal();
00649     game_data_add_score();
00650     allegro_sound_play_effect_goal();
00651
00652     frog_init();
00653
00654     break;
00655
00656   case FROG_STATE_GOAL_COIN:
00657     game_data_add_run_time_goal_bonus();
00658     game_data_add_score_bonus();
00659     allegro_sound_play_effect_bonus();
00660
00661     frog_init();
00662
00663     break;
00664
00665   default:
00666     break;
00667   }
00668
00669 #ifdef DEBUG_ENTITIES_TEXT
00670   if (!(game_frames % 10))
00671     printf("state: %d ~~ y_no_offset: %d\n", frog.state, frog.y - FROG_OFFSET_Y);
00672 #endif
00673 }
00674
00675 static void frog_draw(void)
00676 {
00677
00678   ALLEGRO_BITMAP *tempbitmap = NULL;
00679
00680   if (frog.moving)
00681   {
00682     if (frog.facing == DIRECTION_UP)
00683       tempbitmap = sprites.frog[1];
00684     if (frog.facing == DIRECTION_DOWN)
00685       tempbitmap = sprites.frog[7];
00686     if (frog.facing == DIRECTION_RIGHT)
00687       tempbitmap = sprites.frog[3];
00688     if (frog.facing == DIRECTION_LEFT)
00689       tempbitmap = sprites.frog[5];
00690   }
00691
00692   else if (!frog.moving)
00693   {
00694     if (frog.facing == DIRECTION_UP)
00695       tempbitmap = sprites.frog[0];
00696     if (frog.facing == DIRECTION_DOWN)
00697       tempbitmap = sprites.frog[6];
00698     if (frog.facing == DIRECTION_RIGHT)
00699       tempbitmap = sprites.frog[2];
00700     if (frog.facing == DIRECTION_LEFT)
00701       tempbitmap = sprites.frog[4];
00702   }
00703
00704   al_draw_bitmap(tempbitmap, frog.x, frog.y, 0);
00705
00706 #ifdef DEBUG_ENTITIES_TEXT
00707   // hitbox
00708   allegro_draw_hitbox(frog.x, frog.y, FROG_W, FROG_H);
00709   // coordenadas rana
00710   al_draw_textf(allegro_get_var_font(), al_map_rgb(200, 50, 50), 0, 0, 0, "X: %d Y: %d", frog.x,
      frog.y);
00711 #endif
```

```
00712 }
00713
00714 static void logs_init(void)
00715 {
00716   int i;
00717   for (i = 0; i < LOGS_MAX_USED; i++)
00718     log[i].used = false;
00719 }
00720
00721 static void logs_update(void)
00722 {
00723   // se busca spawnear entre LOGS_SPAWN_MIN y LOGS_SPAWN_MAX autos cada LOGS_SPAWN_FRAMES frames
00724   int new_quota = ((game_frames % LOGS_SPAWN_FRAMES) ? 0 : get_rand_between(LOGS_SPAWN_MIN,
       LOGS_SPAWN_MAX));
00725
00726   int i, used;
00727
00728   // cuento cuantos troncos usados hay
00729   for (i = 0, used = 0; i < LOGS_MAX_USED; i++)
00730     used += log[i].used;
00731
00732   for (i = 0; i < LOGS_MAX_USED; i++)
00733   {
00734     // Spawneo de troncos
00735     if (!log[i].used && new_quota > 0 && used < LOGS_MAX_USED) // Lugar libre?
00736     {
00737
00738       // Asigno carril.
00739       int temp_rand_log_lane = get_rand_between(0, LANES_LOG_TOTAL - 1);
00740       log[i].lane = lanes_logs[temp_rand_log_lane];
00741
00742       // Coordenada 'y' en funcion del carril
00743       log[i].y = CELL_H * log[i].lane + LOG_OFFSET_Y;
00744
00745       // Velocidad
00746       // log[i].dx = lanes_logs[LANES_LOG_TOTAL-1] - log[i].lane + 1;
00747       // log[i].dx = map_int(log[i].lane, 0, lanes_logs[LANES_LOG_TOTAL-1], 1, 3);
00748       log[i].dx = log[i].lane - (temp_rand_log_lane + 2) + LOGS_BASE_SPEED;
00749
00750       // en pares...
00751       if (!(log[i].lane % 2))
00752       {
00753         // coordenada de inicio
00754         log[i].x = -LOG_W;
00755       }
00756
00757       // en impares...
00758       else
00759       {
00760         // coordenada de inicio
00761         log[i].x = DISPLAY_W;
00762
00763         // hacia el otro lado
00764         log[i].dx *= (-1);
00765       }
00766
00767       int p;
00768       bool check; // para confirmar asignacion de lane
00769       for (p = 0, check = true; p < LOGS_MAX_USED; p++)
00770       {
00771         // si no es el mismo tronco, y ese otro esta usado, y coinciden en lane...
00772         if (p != i && log[p].used && log[p].lane == log[i].lane)
00773         {
00774           // si colisiona con algun otro tronco...
00775           if (collide(
00776                   log[i].x - LOGS_EXTRA_SEPARATOR,
00777                   log[i].y,
00778                   log[i].x + LOG_W + LOGS_EXTRA_SEPARATOR,
00779                   log[i].y + LOG_H,
00780                   log[p].x,
00781                   log[p].y,
00782                   log[p].x + LOG_W,
00783                   log[p].y + LOG_H))
00784           {
00785             // no spawnea
00786             check = false;
00787             break;
00788           }
00789         }
00790       }
00791
00792       // si se puede spawnear...
00793       if (check)
00794       {
00795         // Pasa a usado
00796         log[i].used = true;
00797         used++;
```

```
00798         new_quota--;
00799       }
00800
00801       // si no se puede spawnear...
00802       else
00803       {
00804       }
00805     }
00806
00807     // si el tronco esta usado...
00808     else if (log[i].used)
00809     {
00810       // desplaza
00811       log[i].x += log[i].dx;
00812
00813       // chequea si llego a los limites
00814       if ((log[i].dx > 0 && log[i].x >= DISPLAY_W) || (log[i].dx < 0 && log[i].x <= -LOG_W))
00815       {
00816         log[i].used = false;
00817         used--;
00818       }
00819
00820       // printf("~log%d lane%d dx%d~\n", i, log[i].lane, log[i].dx);
00821     }
00822   }
00823 }
00824
00825 static void logs_draw(void)
00826 {
00827   int i;
00828
00829   for (i = 0; i < LOGS_MAX_USED; i++)
00830   {
00831     if (log[i].used)
00832     {
00833       al_draw_bitmap(sprites.log, log[i].x, log[i].y, 0);
00834
00835 #ifdef DEBUG_ENTITIES_TEXT
00836       // hitbox
00837       allegro_draw_hitbox(log[i].x, log[i].y, LOG_W, LOG_H);
00838 #endif
00839     }
00840   }
00841
00842 #ifdef DEBUG_ENTITIES_TEXT
00843   // coordenadas
00844   int space;
00845   for (i = 0, space = 20; i < LOGS_MAX_USED; i++, space += 10)
00846   {
00847     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);
00848   }
00849 #endif
00850 }
00851
00852 static void cars_init(void)
00853 {
00854   int i;
00855   // Inicio array de autos desocupando.
00856   for (i = 0; i < CARS_MAX_USED; i++)
00857     car[i].used = false;
00858
00859   switch (game_data_get_diff())
00860   {
00861   case DIFFICULTIES_EASY:
00862     cars_spawn_max = 3;
00863     break;
00864
00865   case DIFFICULTIES_NORMAL:
00866     normal_diff_lane = get_rand_between(lanes_cars[0], lanes_cars[LANES_CAR_TOTAL - 1]);
00867     cars_spawn_max = 4;
00868     break;
00869
00870   case DIFFICULTIES_HARD:
00871     hard_diff_lane_1 = get_rand_between(lanes_cars[0], lanes_cars[2]);
00872     hard_diff_lane_2 = get_rand_between(lanes_cars[3], lanes_cars[4]);
00873     cars_spawn_max = 5;
00874     break;
00875   }
00876 }
00877
00878 static void cars_update(void)
00879 {
00880   // se busca spawnear entre CARS_SPAWN_MIN y cars_spawn_max autos cada CARS_SPAWN_FRAMES frames
00881   int new_quota = ((game_frames % CARS_SPAWN_FRAMES) ? 0 : get_rand_between(CARS_SPAWN_MIN,
      cars_spawn_max));
00882
```

```
00883   int i, used;
00884
00885   // cuento cuantos autos usados hay
00886   for (i = 0, used = 0; i < CARS_MAX_USED; i++)
00887     used += car[i].used;
00888
00889   for (i = 0; i < CARS_MAX_USED; i++)
00890   {
00891     // Spawneo de autos.
00892     if (!car[i].used && new_quota > 0 && used < CARS_MAX_USED) // Lugar libre?
00893     {
00894       // Asigno carril.
00895       car[i].lane = lanes_cars[get_rand_between(0, LANES_CAR_TOTAL - 1)];
00896
00897       // Coordenada 'y' en funcion del carril
00898       car[i].y = CELL_H * car[i].lane + CAR_OFFSET_Y;
00899
00900       // Velocidad menor en rutas mas alejadas
00901       car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED;
00902       // car[i].dx = CARS_BASE_SPEED;
00903
00904       // Asigno tipos.
00905       car[i].type = get_rand_between(0, CAR_TYPE_N - 1);
00906
00907       // Defino los largos correspondientes,
00908       switch (car[i].type)
00909       {
00910       case CAR_POLICE:
00911       case CAR_YELLOW:
00912       case CAR_BLUE:
00913         car[i].length = CAR_W;
00914         break;
00915       case TRUCK_FIRE:
00916         car[i].length = CAR_TRUCK_FIRE_W;
00917         break;
00918       case TRUCK:
00919         car[i].length = CAR_TRUCK_W;
00920         break;
00921       default:
00922         break;
00923       }
00924
00925       // Inicializo el contador;
00926       car[i].count = 0;
00927
00928       // Inicializo el flag.
00929       car[i].fast = 0;
00930
00931       // en pares...
00932       if (!(car[i].lane % 2))
00933       {
00934         // coordenada de inicio
00935         car[i].x = -car[i].length;
00936       }
00937
00938       // en impares...
00939       else
00940       {
00941         // coordenada de inicio
00942         car[i].x = DISPLAY_W;
00943
00944         // hacia el otro lado
00945         car[i].dx *= (-1);
00946       }
00947
00948       int p;
00949       bool check; // para confirmar asignacion de lane
00950       for (p = 0, check = true; p < CARS_MAX_USED; p++)
00951       {
00952         // si no es el mismo auto, y ese otro esta usado, y coinciden en lane...
00953         if (p != i && car[p].used && car[p].lane == car[i].lane)
00954         {
00955           // si colisiona con algun otro auto...
00956           if (collide(
00957                   car[i].x - CARS_EXTRA_SEPARATOR,
00958                   car[i].y,
00959                   car[i].x + car[i].length + CARS_EXTRA_SEPARATOR, // Es el mas largo.
00960                   car[i].y + CAR_H,
00961                   car[p].x,
00962                   car[p].y,
00963                   car[p].x + car[p].length,
00964                   car[p].y + CAR_H))
00965           {
00966             // no spawnea
00967             check = false;
00968             break;
00969           }
```

```
00970         }
00971       }
00972
00973       // si se puede spawnear...
00974       if (check)
00975       {
00976         // Pasa a usado
00977         car[i].used = true;
00978         used++;
00979         new_quota--;
00980       }
00981
00982       // si no se puede spawnear...
00983       else
00984       {
00985       }
00986     }
00987
00988     // si el auto esta usado...
00989     else if (car[i].used)
00990     {
00991       // Carril con velocidad variable
00992       if (car[i].count < CAR_WAIT_INCREASE)
00993       {
00994         switch (game_data_get_diff())
00995         {
00996         case DIFFICULTIES_EASY:
00997           break;
00998
00999         case DIFFICULTIES_NORMAL:
01000           if (car[i].lane == normal_diff_lane)
01001           {
01002             if (!(game_frames % FPS))
01003             {
01004               if (car[i].fast == 0)
01005               {
01006                 car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED +
      CAR_SPEED_INCREASE;
01007                 if (car[i].lane % 2)
01008                   car[i].dx *= (-1);
01009                 car[i].fast = 1;
01010               }
01011               else
01012               {
01013                 car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED;
01014                 if (car[i].lane % 2)
01015                   car[i].dx *= (-1);
01016                 car[i].fast = 0;
01017               }
01018             }
01019           }
01020           break;
01021         case DIFFICULTIES_HARD:
01022           if ((car[i].lane == hard_diff_lane_1) || (car[i].lane == hard_diff_lane_2))
01023           {
01024             if (!(game_frames % FPS))
01025             {
01026               if (car[i].fast == 0)
01027               {
01028                 car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED +
      CAR_SPEED_INCREASE;
01029                 if (car[i].lane % 2)
01030                   car[i].dx *= (-1);
01031                 car[i].fast = 1;
01032               }
01033               else
01034               {
01035                 car[i].dx = car[i].lane - (MAX_LANES - LANES_CAR_TOTAL) + CARS_BASE_SPEED;
01036                 if (car[i].lane % 2)
01037                   car[i].dx *= (-1);
01038                 car[i].fast = 0;
01039               }
01040             }
01041           }
01042         default:
01043           break;
01044         }
01045       }
01046       else
01047         car[i].count++;
01048
01049       // Desplazamiento
01050       car[i].x += car[i].dx;
01051
01052       // chequea si llego a los limites
01053       if ((car[i].dx > 0 && car[i].x >= DISPLAY_W) || (car[i].dx < 0 && car[i].x <= -car[i].length))
01054       {
```

```
01055         car[i].used = false;
01056         used--;
01057       }
01058
01059       // printf("~car%d lane%d dx%d~\n", i, car[i].lane, car[i].dx);
01060     }
01061   }
01062 }
01063
01064 static void cars_draw()
01065 {
01066   int i;
01067   bool flag;
01068
01069   ALLEGRO_BITMAP *temp_bitmap = NULL;
01070
01071   for (i = 0; i < CARS_MAX_USED; i++)
01072   {
01073     if (car[i].used)
01074     {
01075       if (car[i].dx < 0)
01076         flag = ALLEGRO_FLIP_HORIZONTAL;
01077       else
01078         flag = 0;
01079
01080       temp_bitmap = sprites.car[car[i].type];
01081
01082       // Dibujo los autos en sus carriles.
01083       al_draw_bitmap(temp_bitmap, car[i].x, car[i].y, flag);
01084
01085 #ifdef DEBUG_ENTITIES_TEXT
01086       // Dibujo hitbox
01087       allegro_draw_hitbox(car[i].x, car[i].y, car[i].length, CAR_H);
01088 #endif
01089     }
01090   }
01091
01092 #ifdef DEBUG_ENTITIES_TEXT
01093   // coordenadas
01094   int space;
01095   for (i = 0, space = 200; i < CARS_MAX_USED; i++, space += 20)
01096   {
01097     // 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);
01098     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);
01099   }
01100 #endif
01101 }
01102
01103 static void turtles_init(void)
01104 {
01105   int i;
01106   for (i = 0; i < TURTLES_MAX_USED; i++)
01107   {
01108     turtle_pack[i].used = false;
01109   }
01110 }
01111
01112 static void turtles_update(void)
01113 {
01114   int new_quota = ((game_frames % TURTLES_SPAWN_FRAMES) ? 0 : get_rand_between(TURTLES_SPAWN_MIN,
      TURTLES_SPAWN_MAX));
01115
01116   int i, used;
01117
01118   for (i = 0, used = 0; i < TURTLES_MAX_USED; i++)
01119   {
01120     if (turtle_pack[i].used)
01121       used++;
01122   }
01123
01124   for (i = 0; i < TURTLES_MAX_USED; i++)
01125   {
01126     // Spawneo de turtle_packs
01127     if (!turtle_pack[i].used && new_quota > 0 && used < TURTLES_MAX_USED) // Lugar libre?
01128     {
01129
01130       // defino tortugas en el pack
01131       turtle_pack[i].turtles_in_pack = get_rand_between(TURTLES_MIN_PER_PACK, TURTLES_MAX_PER_PACK);
01132
01133       // calculo ancho del pack
01134       turtle_pack[i].wide = TURTLE_SIDE * turtle_pack[i].turtles_in_pack;
01135
01136       // Asigno carril.
01137       turtle_pack[i].lane = lanes_turtles[get_rand_between(0, LANES_TURTLE_TOTAL - 1)];
01138
```

```
01139       // Coordenada 'y' en funcion del carril
01140       turtle_pack[i].y = CELL_H * turtle_pack[i].lane;
01141
01142       // Velocidad
01143       // turtle_pack[i].dx = lanes_turtles[LANES_TURTLE_TOTAL- turtle_pack[i].lane + 1];
01144       turtle_pack[i].dx = TURTLES_BASE_SPEED;
01145
01146       // en pares...
01147       if (!(turtle_pack[i].lane % 2))
01148       {
01149         // coordenada de inicio
01150         turtle_pack[i].x = -turtle_pack[i].wide;
01151       }
01152
01153       // en impares...
01154       else
01155       {
01156         // coordenada de inicio
01157         turtle_pack[i].x = DISPLAY_W;
01158
01159         // hacia el otro lado
01160         turtle_pack[i].dx *= (-1);
01161       }
01162
01163       int p;
01164       bool check; // para confirmar asignacion de lane
01165       for (p = 0, check = true; p < TURTLES_MAX_USED; p++)
01166       {
01167         // si no es el mismo pack, y ese otro esta usado, y coinciden en lane...
01168         if (p != i && turtle_pack[p].used && turtle_pack[p].lane == turtle_pack[i].lane)
01169         {
01170           // si colisiona con algun otro pack...
01171           if (collide(
01172                   turtle_pack[i].x - TURTLES_EXTRA_SEPARATOR,
01173                   turtle_pack[i].y,
01174                   turtle_pack[i].x + turtle_pack[i].wide + TURTLES_EXTRA_SEPARATOR,
01175                   turtle_pack[i].y + TURTLE_SIDE,
01176                   turtle_pack[p].x,
01177                   turtle_pack[p].y,
01178                   turtle_pack[p].x + turtle_pack[p].wide,
01179                   turtle_pack[p].y + TURTLE_SIDE))
01180           {
01181             // no spawnea
01182             check = false;
01183             break;
01184           }
01185         }
01186       }
01187
01188       // si se puede spawnear...
01189       if (check)
01190       {
01191         // Pasa a usado
01192         turtle_pack[i].used = true;
01193         used++;
01194
01195         // se inicializa el contador de frames
01196         turtle_pack[i].fx.frame = 0;
01197         turtle_pack[i].fx.cont = 1;
01198         turtle_pack[i].fx.timeout = 0;
01199         // fuera del agua
01200         turtle_pack[i].state = TURTLE_STATE_SURFACE;
01201
01202         new_quota--;
01203       }
01204
01205       // si no se puede spawnear...
01206       else
01207       {
01208       }
01209     }
01210
01211     // si el pack esta usado...
01212     else if (turtle_pack[i].used)
01213     {
01214       // desplaza
01215       turtle_pack[i].x += turtle_pack[i].dx;
01216
01217       switch (turtle_pack[i].state)
01218       {
01219       case TURTLE_STATE_SURFACE:
01220         if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_SURFACE))
01221           turtle_pack[i].fx.frame++;
01222
01223         // si no esta inicializado, inicializo timeout
01224         if (!turtle_pack[i].fx.timeout)
01225           turtle_pack[i].fx.timeout = get_rand_between(TURTLES_SURFACE_FRAMES_MIN,
```

```
      TURTLES_SURFACE_FRAMES_MAX);
01226
01227         // pasa a agua
01228         if (!(turtle_pack[i].fx.cont % turtle_pack[i].fx.timeout))
01229         {
01230           turtle_pack[i].state = TURTLE_STATE_GOING_DOWN;
01231           turtle_pack[i].fx.frame = 7;
01232           turtle_pack[i].fx.timeout = 0;
01233           turtle_pack[i].fx.cont = 1;
01234         }
01235
01236         // Reinicia animacion
01237         else if (turtle_pack[i].fx.frame == 7)
01238           turtle_pack[i].fx.frame = 0;
01239
01240         break;
01241
01242       case TURTLE_STATE_GOING_DOWN:
01243         if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_GOING_DOWN))
01244           turtle_pack[i].fx.frame++;
01245
01246         if (turtle_pack[i].fx.frame == 9)
01247         {
01248           turtle_pack[i].state = TURTLE_STATE_WATER;
01249           turtle_pack[i].fx.cont = 1;
01250         }
01251
01252         break;
01253
01254       case TURTLE_STATE_WATER:
01255         if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_WATER))
01256           turtle_pack[i].fx.frame++;
01257
01258         // si no esta inicializado, inicializo timeout
01259         if (!turtle_pack[i].fx.timeout)
01260           turtle_pack[i].fx.timeout = get_rand_between(TURTLES_WATER_FRAMES_MIN,
      TURTLES_WATER_FRAMES_MAX);
01261
01262         // pasa a fuera
01263         if (!(turtle_pack[i].fx.cont % turtle_pack[i].fx.timeout))
01264         {
01265           turtle_pack[i].state = TURTLE_STATE_GOING_UP;
01266           turtle_pack[i].fx.frame = 10;
01267           turtle_pack[i].fx.timeout = 0;
01268           turtle_pack[i].fx.cont = 1;
01269         }
01270
01271         // Reinicia animacion
01272         else if (turtle_pack[i].fx.frame == 11)
01273           turtle_pack[i].fx.frame = 9;
01274
01275         break;
01276
01277       case TURTLE_STATE_GOING_UP:
01278         if (!(turtle_pack[i].fx.cont++ % TURTLES_FRAME_TIMEOUT_GOING_UP))
01279           turtle_pack[i].fx.frame--;
01280
01281         if (turtle_pack[i].fx.frame == 7)
01282         {
01283           turtle_pack[i].fx.frame = 6;
01284           turtle_pack[i].state = TURTLE_STATE_SURFACE;
01285           turtle_pack[i].fx.cont = 1;
01286         }
01287
01288         break;
01289
01290       default:
01291         break;
01292       }
01293
01294       // chequea si llego a los limites
01295       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))
01296       {
01297         turtle_pack[i].used = false;
01298         used--;
01299       }
01300
01301       // printf("~turtle_pack%d lane%d dx%d~\n", i, turtle_pack[i].lane, turtle_pack[i].dx);
01302     }
01303   }
01304 }
01305
01306 static void turtles_draw(void)
01307 {
01308   int i, j, flag;
01309   for (i = 0; i < TURTLES_MAX_USED; i++)
```

```
01310   {
01311     if (turtle_pack[i].used)
01312     {
01313       for (j = 0; j < turtle_pack[i].turtles_in_pack; j++)
01314       {
01315         if (turtle_pack[i].dx < 0)
01316           flag = ALLEGRO_FLIP_HORIZONTAL;
01317         else
01318           flag = 0;
01319 
01320         al_draw_bitmap(sprites.turtle[turtle_pack[i].fx.frame], turtle_pack[i].x + TURTLE_SIDE * j,
       turtle_pack[i].y, flag);
01321       }
01322 
01323 #ifdef DEBUG_ENTITIES_TEXT
01324       // Dibujo hitbox
01325       allegro_draw_hitbox(turtle_pack[i].x, turtle_pack[i].y, turtle_pack[i].wide, TURTLE_SIDE);
01326 #endif
01327     }
01328   }
01329 
01330 #ifdef DEBUG_ENTITIES_TEXT
01331   // coordenadas
01332   int space;
01333   for (i = 0, space = 350; i < TURTLES_MAX_USED; i++, space += 10)
01334   {
01335     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);
01336   }
01337 #endif
01338 }
01339 
01340 static void coin_init(void)
01341 {
01342   coin.used = false;
01343   coin.y = CELL_H + SPRITE_COIN_OFFSET_XY + GOAL_ROW_OFFSET_Y_FIX;
01344 
01345   coin.fx.blink_timer = 0;
01346   coin.fx.timeout = 0;
01347   coin.fx.flag = false;
01348   coin.fx.cont = 1;
01349 }
01350 
01351 static void coin_update(void)
01352 {
01353   if (!coin.used)
01354   {
01355     // si no esta inicializado, inicializo timeout para spawneo
01356     if (!coin.fx.timeout)
01357       coin.fx.timeout = get_rand_between(COIN_SPAWN_FRAMES_MIN, COIN_SPAWN_FRAMES_MAX);
01358 
01359     if (!(coin.fx.cont % coin.fx.timeout))
01360     {
01361       // calculo de coordenada x para alguno de los puntos de llegada
01362       int temp_goal = get_rand_between(0, MAX_GOALS - 1);
01363 
01364       // si el goal está libre...
01365       if (!game_data_get_goal_state(temp_goal))
01366       {
01367         allegro_sound_play_effect_coin_drop();
01368 
01369         coin.x = CELL_W * goal_cols[temp_goal] + SPRITE_COIN_OFFSET_XY - 1;
01370         // marcado como usado
01371         coin.used = true;
01372         // desinicializo el timeout
01373         coin.fx.timeout = 0;
01374 
01375         coin.fx.blink_timer = 0;
01376         coin.fx.cont = 1;
01377         coin.fx.frame_cont = 0;
01378       }
01379 
01380       // si no, cuando pasa otro timeout se intenta de nuevo
01381       else
01382       {
01383       }
01384     }
01385   }
01386 
01387   else
01388   {
01389     // timeout para despawneo
01390     if (!coin.fx.timeout)
01391       coin.fx.timeout = get_rand_between(COIN_DESPAWN_FRAMES_MIN, COIN_DESPAWN_FRAMES_MAX);
01392 
01393     if (++coin.fx.blink_timer > coin.fx.timeout - COIN_FRAMES_TO_WARN_A)
01394     {
```

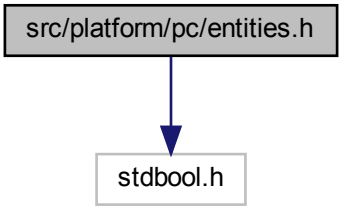
```
01395       if (coin.fx.blink_timer > coin.fx.timeout - COIN_FRAMES_TO_WARN_B)
01396       {
01397         if (!(coin.fx.cont % COIN_WARNING_FRAMES_B))
01398           coin.fx.flag = !coin.fx.flag;
01399       }
01400       else
01401       {
01402         if (!(coin.fx.cont % COIN_WARNING_FRAMES_A))
01403           coin.fx.flag = !coin.fx.flag;
01404       }
01405     }
01406
01407     if (!(game_frames % COIN_FRAME_RATE))
01408     {
01409       if (++coin.fx.frame_cont == SPRITE_COIN_FRAMES)
01410         coin.fx.frame_cont = 0;
01411     }
01412
01413     // si se puede despawnear
01414     if (!(coin.fx.cont % coin.fx.timeout))
01415     {
01416       // coin no usada
01417       coin.used = false;
01418
01419       // desinicializo timeout
01420       coin.fx.timeout = 0;
01421
01422       // saco el blinking
01423       coin.fx.flag = false;
01424
01425       coin.fx.cont = 1;
01426     }
01427   }
01428
01429   coin.fx.cont++;
01430 }
01431
01432 static void coin_draw(void)
01433 {
01434   if (coin.used)
01435   {
01436     // Si no está el flag, dibujo sprite normalmente
01437     if (!coin.fx.flag)
01438       al_draw_bitmap(sprites.coin.frame[coin.fx.frame_cont], coin.x, coin.y, 0);
01439   }
01440
01441 #ifdef DEBUG_ENTITIES_TEXT
01442
01443   // hitbox
01444   allegro_draw_hitbox(coin.x, coin.y, COIN_SIDE, COIN_SIDE);
01445
01446   // coordenadas
01447   int space = 500;
01448   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);
01449 #endif
01450 }
01451
01452 /*
01453 static void fix_frog_pos(void)
01454 {
01455   //coordenadas topleft, sin offset
01456   //x entre (0) y (DISPLAY_W - CELL_W)
01457   //y entre (CELL_H) y (DISPLAY_H - CELL_H)
01458   int x = (frog.x - FROG_OFFSET_X), y = (frog.y - FROG_OFFSET_Y);
01459
01460   //coordenadas topleft "correctas"
01461   int x_values[COLS], y_values[ROWS];
01462
01463   int i;
01464
01465   for(i = 0; i < COLS; i++)
01466     x_values[i] = i * CELL_W;
01467   for(i = 1; i < ROWS; i++)
01468     y_values[i] = i * CELL_H;
01469
01470   int temp_a, temp_b;
01471
01472   for(i = 0; i < COLS; i++)
01473   {
01474     temp_a = x - x_values[i];
01475
01476     if(temp_a > 0)
01477       continue;
01478     if(temp_a == 0)
01479       break;
01480
```

```
01481     temp_b = x_values[i-1] - x;
01482
01483
01484     if((i-1) >= 0)
01485     {
01486       //"si está más cerca de la columna 'i' que de la 'i+1"
01487       if(temp_a <= temp_b)
01488         frog.x = x_values[i-1] + FROG_OFFSET_X;
01489       else
01490         frog.x = x_values[i] + FROG_OFFSET_X;
01491     }
01492
01493     break;
01494   }
01495
01496   for(i = 1; i < ROWS; i++)
01497   {
01498     temp_a = y - y_values[i];
01499
01500     if(temp_a > 0)
01501       continue;
01502     if(temp_a == 0)
01503       break;
01504
01505     temp_b = y_values[i-1] - y;
01506
01507     //"si está más cerca de la fila 'i' que de la 'i+1"
01508     if(temp_a <= temp_b)
01509       frog.y = y_values[i-1] + FROG_OFFSET_Y;
01510     else
01511       frog.y = y_values[i] + FROG_OFFSET_Y;
01512
01513     break;
01514   }
01515 }
01516 */
01517
01518 static void fix_frog_coord_y(void)
01519 {
01520   int y = (frog.y - FROG_OFFSET_Y);
01521
01522   int y_values[ROWS];
01523
01524   int i;
01525
01526   // Carga valores "correctos" de y
01527   for (i = 1; i < ROWS - 1; i++)
01528     y_values[i] = i * CELL_H;
01529
01530   int temp_a, temp_b;
01531   for (i = 1; i < ROWS - 1; i++)
01532   {
01533     temp_a = y - y_values[i];
01534
01535     if (temp_a > 0)
01536       continue;
01537     if (temp_a == 0)
01538       break;
01539
01540     temp_b = y_values[i - 1] - y;
01541
01542     //"si está más cerca de la fila 'i' que de la 'i+1"
01543     if (temp_a <= temp_b)
01544       frog.y = y_values[i - 1] + FROG_OFFSET_Y;
01545     else
01546       frog.y = y_values[i] + FROG_OFFSET_Y;
01547
01548     break;
01549   }
01550 }
01551
01552 static bool is_frog_in_goal(void)
01553 {
01554   bool state = false;
01555   int x = frog.x;
01556
01557   int i, x_col;
01558   for (i = 0; i < MAX_GOALS; i++)
01559   {
01560     // Coordenada top left del punto de llegada
01561     x_col = goal_cols[i] * CELL_W;
01562
01563     // Calculo para ver si entro bien o no
01564     if ((x > x_col - GOAL_ROW_MARGIN_TO_REACH) &&
01565         ((x + FROG_W) < x_col + CELL_W + GOAL_ROW_MARGIN_TO_REACH))
01566     {
01567       // coodenada X aceptable
```

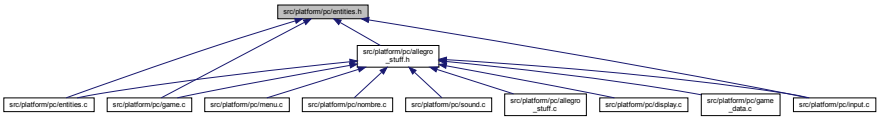
```
01568       state = true;
01569       break;
01570     }
01571   }
01572
01573   // Si coincide en coordenada y el goal esta libre...
01574   if (state && !game_data_get_goal_state(i))
01575   {
01576     // marca el goal como completo
01577     game_data_set_goal(i);
01578   }
01579   else
01580   {
01581     // no llego a un goal valido
01582     state = false;
01583   }
01584
01585   return state;
01586 }
01587
01588 static void corpse_init(int x, int y)
01589 {
01590   corpse_fx.flag = true;
01591   corpse_fx.timer = 1;
01592   corpse_fx.x = x - FROG_OFFSET_X + SPRITE_DEAD_OFFSET;
01593   corpse_fx.y = y - FROG_OFFSET_Y + SPRITE_DEAD_OFFSET;
01594 }
01595
01596 static void corpse_update(void)
01597 {
01598   if (corpse_fx.flag)
01599   {
01600     if (!(corpse_fx.timer++ % SPRITE_DEAD_TIMEOUT))
01601       corpse_fx.flag = false;
01602   }
01603 }
01604
01605 static void corpse_draw(void)
01606 {
01607   if (corpse_fx.flag)
01608     al_draw_bitmap(sprites.dead, corpse_fx.x, corpse_fx.y, 0);
01609 }
01610
01611 static void splash_init(int x, int y)
01612 {
01613   splash_fx.flag = true;
01614   splash_fx.cont = 1;
01615   splash_fx.frame_cont = 0;
01616   splash_fx.x = x - FROG_OFFSET_X + SPRITE_SPLASH_OFFSET_X;
01617   splash_fx.y = y - FROG_OFFSET_Y + SPRITE_SPLASH_OFFSET_Y;
01618 }
01619
01620 static void splash_update(void)
01621 {
01622   if (splash_fx.flag)
01623   {
01624     if (!(splash_fx.cont % SPRITE_SPLASH_RATE))
01625     {
01626       if (++splash_fx.frame_cont == SPRITE_SPLASH_FRAMES)
01627       {
01628         splash_fx.frame_cont = 0;
01629         splash_fx.flag = false;
01630       }
01631     }
01632
01633     splash_fx.cont++;
01634   }
01635 }
01636
01637 static void splash_draw(void)
01638 {
01639   if (splash_fx.flag)
01640     al_draw_bitmap(sprites.splash.frame[splash_fx.frame_cont], splash_fx.x, splash_fx.y, 0);
01641 }
```

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

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



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



Functions

- void entities_init (void)

  Inicializa las entidades.
- void entities_update (void)

  Actualiza las entidades.
- void entities_draw (void)

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

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

4.32.1 Detailed Description

Author

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file entities.h.

4.32.2 Function Documentation

4.32.2.1 entities_draw() `void entities_draw (`

`void )`

Dibuja las entidades.

Definition at line 400 of file entities.c.

4.32.2.2 entities_init() `void entities_init (`

`void )`

Inicializa las entidades.

Definition at line 371 of file entities.c.

4.32.2.3 entities_move_frog() `void entities_move_frog (`

`unsigned char direction )`

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

Parameters

direction	enum DIRECTIONS

Definition at line 414 of file entities.c.

4.32.2.4 entities_update() `void entities_update (`

`void )`

Actualiza las entidades.

Definition at line 386 of file entities.c.

4.33 entities.h

Go to the documentation of this file.

```
00001
00010 #ifndef _ENTITIES_H_
00011 #define _ENTITIES_H_
00012
00013 /*****************************************************************************
00014  * INCLUDE HEADER FILES
```

```
00015  ******************************************************************************/
00016
00017 #include <stdbool.h>
00018
00019 /******************************************************************************
00020  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00021  ******************************************************************************/
00022
00027 void entities_init(void);
00028
00033 void entities_update(void);
00034
00039 void entities_draw(void);
00040
00046 void entities_move_frog(unsigned char direction);
00047
00048 /******************************************************************************
00049  ******************************************************************************/
00050
00051 #endif // _ENTITIES_H_
```

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

```
#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.

4.34.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-22

Copyright

Definition in file game.c.

4.34.2 Function Documentation

4.34.2.1 actualizarInterfaz() `void actualizarInterfaz ( void )`

Actualizaciones relativas a lo visual.

Definition at line 155 of file game.c.

4.34.2.2 getMaxPuntos() `unsigned long long getMaxPuntos ( void )`

Devuelve el puntaje máximo del jugador.

Returns

unsigned long long

Definition at line 82 of file game.c.

4.34.2.3 getNivel() `int getNivel ( void )`

Devuelve el nivel//run del jugador.

Returns

int

Definition at line 87 of file game.c.

4.34.2.4 getNombre() `char * getNombre ( void )`

Devuelve el nombre del jugador.

Returns

char*

Definition at line 72 of file game.c.

4.34.2.5 getPuntos() unsigned long long getPuntos (void)

Devuelve el puntaje del jugador.

Returns

unsigned long long

Definition at line 77 of file game.c.

4.34.2.6 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.

4.34.2.7 moverAdelante() void moverAdelante (void)

Avanza el jugador.

Definition at line 131 of file game.c.

4.34.2.8 moverAtras() void moverAtras (void)

Retrocede el jugador.

Definition at line 136 of file game.c.

4.34.2.9 moverDcha() void moverDcha (void)

Mueve el jugador a la derecha.

Definition at line 146 of file game.c.

4.34.2.10 moverIzda() `void moverIzda (`

`void )`

Mueve el jugador a la izquierda.

Definition at line 141 of file game.c.

4.34.2.11 pausarJuego() `void pausarJuego (`

`void )`

Pausa el juego.

Definition at line 101 of file game.c.

4.34.2.12 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 188 of file game.c.

4.34.2.13 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 114 of file game.c.

4.34.2.14 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 105 of file game.c.

4.34.2.15 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 151 of file game.c.

4.34.2.16 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 47 of file game.c.

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

Setea los puntos maximos del jugador.

Parameters

max	

Definition at line 42 of file game.c.

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

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 37 of file game.c.

4.34.2.19 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.

4.35 game.c

Go to the documentation of this file.

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

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

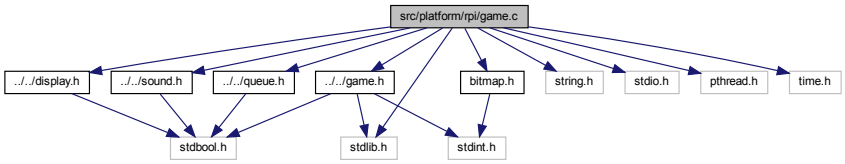
```
00181
00182     al_flip_display();
00183
00184     allegro_set_var_redraw(false);
00185   }
00186 }
00187
00188 void reanudarJuego(void)
00189 {
00190 }
```

4.36 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 <stdio.h>
#include <pthread.h>
#include <time.h>
```

Include dependency graph for game.c:



Macros

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

Functions

- void setNombre (char *nombre)

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

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

  Actualizaciones relativas a actualizar las entidades.

- bool tiempoRefrescoEntidades (void)

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

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

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

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

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

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

  Configuraciones para reiniciar el nivel.
- void respawn ()

  Respawnea el jugador.
- void moverAdelante ()

  Avanza el jugador.
- void moverAtras ()

  Retrocede el jugador.
- void moverIzda ()

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

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

  Resta una vida.
- void inicializarJuego ()

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

  Pausa el juego.
- void actualizarInterfaz ()

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

  Saca el juego de pausa.

Variables

- matriz_t disp_matriz

4.36.1 Detailed Description

Archivo para manejar la información del juego.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file game.c.

4.36.2 Macro Definition Documentation

4.36.2.1 CANT_CARRILES `#define CANT_CARRILES 5`

Definition at line 31 of file game.c.

4.36.2.2 L_MAX `#define L_MAX 64`

Definition at line 32 of file game.c.

4.36.2.3 POS_AUTOS_FIN `#define POS_AUTOS_FIN 13`

Definition at line 30 of file game.c.

4.36.2.4 POS_AUTOS_INICIO `#define POS_AUTOS_INICIO 4`

Definition at line 29 of file game.c.

4.36.3 Function Documentation

4.36.3.1 actualizarInterfaz() `void actualizarInterfaz ( void )`

Actualizaciones relativas a lo visual.

Definition at line 342 of file game.c.

4.36.3.2 actualizarMapa() `void actualizarMapa ( )`

Definition at line 149 of file game.c.

4.36.3.3 getMaxPuntos() `unsigned long long getMaxPuntos ( void )`

Devuelve el puntaje máximo del jugador.

Returns

unsigned long long

Definition at line 204 of file game.c.

4.36.3.4 getNivel() `int getNivel ( void )`

Devuelve el nivel//run del jugador.

Returns

int

Definition at line 209 of file game.c.

4.36.3.5 getNombre() `char * getNombre ( void )`

Devuelve el nombre del jugador.

Returns

char*

Definition at line 195 of file game.c.

4.36.3.6 getPuntos() `unsigned long long getPuntos ( void )`

Devuelve el puntaje del jugador.

Returns

unsigned long long

Definition at line 199 of file game.c.

4.36.3.7 inicializarJuego() `void inicializarJuego ( void )`

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

Definition at line 315 of file game.c.

4.36.3.8 limpiarMapa() `void limpiarMapa ( )`

Definition at line 117 of file game.c.

4.36.3.9 moverAdelante() `void moverAdelante ( void )`

Avanza el jugador.

Definition at line 246 of file game.c.

4.36.3.10 moverAtras() `void moverAtras ( void )`

Retrocede el jugador.

Definition at line 276 of file game.c.

4.36.3.11 moverCarrriles() `void moverCarrriles ( int x )`

Definition at line 121 of file game.c.

4.36.3.12 moverDcha() `void moverDcha ( void )`

Mueve el jugador a la derecha.

Definition at line 292 of file game.c.

4.36.3.13 moverIzda() `void moverIzda (`

`void )`

Mueve el jugador a la izquierda.

Definition at line 282 of file game.c.

4.36.3.14 pausarJuego() `void pausarJuego (`

`void )`

Pausa el juego.

Definition at line 334 of file game.c.

4.36.3.15 perderVida() `void perderVida (`

`void )`

Resta una vida.

Definition at line 302 of file game.c.

4.36.3.16 reanudarJuego() `void reanudarJuego (`

`void )`

Saca el juego de pausa.

Definition at line 365 of file game.c.

4.36.3.17 refrescar() `void refrescar (`

`void )`

Actualizaciones relativas a actualizar las entidades.

Definition at line 168 of file game.c.

4.36.3.18 reiniciarNivel() `void reiniciarNivel (`

`void )`

Configuraciones para reiniciar el nivel.

Definition at line 214 of file game.c.

4.36.3.19 respawn() `void respawn (`

`void )`

Respawnea el jugador.

Definition at line 223 of file game.c.

4.36.3.20 setDificultad() `void setDificultad (`

`int dif )`

Setea la dificultad a usar.

Parameters

dif	

Definition at line 190 of file game.c.

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

Setea los puntos maximos del jugador.

Parameters

max	

Definition at line 112 of file game.c.

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

Confirma el nombre del jugador.

Parameters

nombre	

Definition at line 107 of file game.c.

4.36.3.23 spawnearAutos() `void spawnearAutos ( )`

Definition at line 127 of file game.c.

4.36.3.24 tiempoRefrescoEntidades() `bool tiempoRefrescoEntidades ( void )`

Chequea si es tiempo de refrescar entidades según la plataforma.

Returns

true

false

Definition at line 185 of file game.c.

4.36.4 Variable Documentation

4.36.4.1 agua `bool agua`

Definition at line 52 of file game.c.

4.36.4.2 completo `uint32_t completo`

Definition at line 64 of file game.c.

4.36.4.3 derecho `uint16_t derecho`

Definition at line 67 of file game.c.

4.36.4.4 dificultad `int dificultad`

Definition at line 41 of file game.c.

4.36.4.5 disp_matriz `matriz_t disp_matriz [extern]`

Definition at line 38 of file display.c.

4.36.4.6 izquierdo `uint16_t izquierdo`

Definition at line 68 of file game.c.

4.36.4.7 jugador_1 `uint16_t jugador_1`

Definition at line 47 of file game.c.

4.36.4.8 jugador_2 `uint16_t jugador_2`

Definition at line 48 of file game.c.

4.36.4.9 jugador_posicion_oeste `int jugador_posicion_oeste`

Definition at line 44 of file game.c.

4.36.4.10 jugador_posicion_sur `int jugador_posicion_sur`

Definition at line 43 of file game.c.

4.36.4.11 jugando `bool jugando`

Definition at line 51 of file game.c.

4.36.4.12 mapa `matriz_t mapa`

Definition at line 61 of file game.c.

4.36.4.13 max_puntos `unsigned long long max_puntos`

Definition at line 50 of file game.c.

4.36.4.14 niv_actual `int niv_actual`

Definition at line 42 of file game.c.

4.36.4.15 nombre_jugador `char nombre_jugador[L_MAX]`

Definition at line 40 of file game.c.

4.36.4.16 puntos `unsigned long long puntos`

Definition at line 49 of file game.c.

4.36.4.17 ranas `uint16_t ranas`

Definition at line 46 of file game.c.

4.36.4.18 refresco_autos `bool refresco_autos`

Definition at line 54 of file game.c.

4.36.4.19 refresco_jugador `bool refresco_jugador`

Definition at line 53 of file game.c.

4.36.4.20 tiempo `clock_t tiempo`

Definition at line 56 of file game.c.

4.36.4.21 tiempo_inicio `clock_t tiempo_inicio`

Definition at line 57 of file game.c.

4.36.4.22 tiempo_referencia `clock_t tiempo_referencia`

Definition at line 58 of file game.c.

4.36.4.23 tiempo_refresco_autos `clock_t tiempo_refresco_autos`

Definition at line 60 of file game.c.

4.36.4.24 tiempo_refresco_jugador `clock_t tiempo_refresco_jugador`

Definition at line 59 of file game.c.

4.36.4.25 timeout `bool timeout`

Definition at line 55 of file game.c.

4.36.4.26 vidas `uint16_t vidas`

Definition at line 45 of file game.c.

4.37 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 <stdio.h>
00022 #include <pthread.h>
00023 #include <time.h>
00024
00025 /*******************************************************************************
00026  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00027  ******************************************************************************/
00028
00029 #define POS_AUTOS_INICIO 4
00030 #define POS_AUTOS_FIN 13
00031 #define CANT_CARRILES 5
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 timeout;
00056   clock_t tiempo;
00057   clock_t tiempo_inicio;
00058   clock_t tiempo_referencia;
00059   clock_t tiempo_refresco_jugador;
```

```
00060   clock_t tiempo_refresco_autos;
00061   matriz_t mapa;
00062   union
00063   {
00064     uint32_t completo;
00065     struct
00066     {
00067       uint16_t derecho;
00068       uint16_t izquierdo;
00069     };
00070   } carril[CANT_CARRILES];
00071 } juego;
00072
00073 /*******************************************************************************
00074  * VARIABLES WITH GLOBAL SCOPE
00075  ******************************************************************************/
00076
00077 extern matriz_t disp_matriz;
00078
00079 /*******************************************************************************
00080  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00081  ******************************************************************************/
00082
00083 static void reiniciarTimer();
00084
00085 /*******************************************************************************
00086  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00087  ******************************************************************************/
00088
00089 // +ej: static const int temperaturas_medias[4] = {23, 26, 24, 29};+
00090
00091 /*******************************************************************************
00092  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00093  ******************************************************************************/
00094
00095 static pthread_t ttiempo, tjugador, tautos;
00096
00097 static void *threadTiempo(void *ptr);
00098 static void *threadJugador(void *ptr);
00099 static void *threadAutos(void *ptr);
00100
00101 /*******************************************************************************
00102  *******************************************************************************
00103                         GLOBAL FUNCTION DEFINITIONS
00104  *******************************************************************************
00105  ******************************************************************************/
00106
00107 void setNombre(char *nombre)
00108 {
00109   strcpy(juego.nombre_jugador, nombre);
00110 }
00111
00112 void setMaxPuntos(unsigned long long max)
00113 {
00114   juego.max_puntos = max;
00115 }
00116
00117 void limpiarMapa()
00118 {
00119 }
00120
00121 void moverCarrriles(int x)
00122 {
00123   for (int i = 0; i < 5; i++)
00124     juego.carril[i].completo «= x;
00125 }
00126
00127 void spawnearAutos()
00128 {
00129   int i;
00130   for (i = 0; i < 5; i++)
00131   {
00132     if (juego.agua)
00133     {
00134       if (!(juego.carril[i].completo & 0b1111111111) && !(rand() % 10))
00135         juego.carril[i].completo |= 0b111111;
00136       else if (!(juego.carril[i].completo & 0b11111111111) && !(rand() % 20))
00137         juego.carril[i].completo |= 0b11111111;
00138     }
00139     else
00140     {
00141       if (!(juego.carril[i].completo & 0b111111) && !(rand() % 10))
00142         juego.carril[i].completo |= 0b11;
00143       else if (!(juego.carril[i].completo & 0b11111111) && !(rand() % 20))
00144         juego.carril[i].completo |= 0b1111;
00145     }
00146   }
```

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

```
00234     juego.carril[i].completo = 0;
00235   limpiarMatriz(juego.mapa);
00236
00237   for (int i = 0; i < 10; i++)
00238   {
00239     moverCarrriles(4);
00240     spawnearAutos();
00241   }
00242
00243   actualizarMapa();
00244 }
00245
00246 void moverAdelante()
00247 {
00248   if (juego.jugador_posicion_sur > 3)
00249     juego.jugador_posicion_sur--;
00250   if (juego.jugador_posicion_sur == 3)
00251   {
00252     if (!juego.agua)
00253     {
00254       juego.agua = true;
00255       respawn();
00256     }
00257     else
00258     {
00259       juego.ranas |= juego.jugador_1 | juego.jugador_2;
00260       if (juego.ranas == 0b1111111111111111)
00261       {
00262         juego.timeout = true;
00263         juego.niv_actual++;
00264         reproducirEfecto(EFECTO_NIVEL_COMPLETO);
00265         reiniciarNivel();
00266       }
00267       else
00268       {
00269         juego.agua = false;
00270         respawn();
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 && !juego.timeout ? 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   reiniciarTimer();
00313 }
00314
00315 void inicializarJuego()
00316 {
00317   juego.puntos = 0;
00318   juego.niv_actual = 1;
00319   juego.vidas = 0b1111100000000000;
00320 }
```

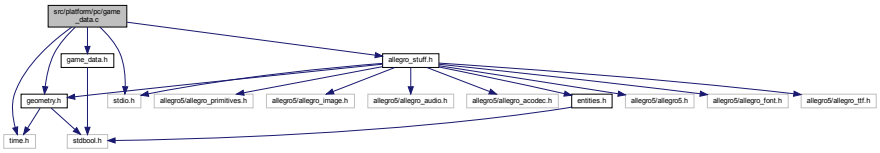
```
00321
00322 void reiniciarTimer()
00323 {
00324   juego.tiempo_inicio = CLOCKS_PER_SEC * 60 - (juego.dificultad - juego.niv_actual);
00325   juego.tiempo = juego.tiempo_inicio;
00326   juego.tiempo_referencia = juego.tiempo_inicio;
00327   juego.tiempo_refresco_autos = CLOCKS_PER_SEC * (1 - 0.125 * (juego.dificultad - juego.niv_actual -
      1));
00328   juego.tiempo_refresco_jugador = CLOCKS_PER_SEC » 1;
00329   juego.refresco_autos = false;
00330   juego.refresco_jugador = false;
00331   juego.timeout = false;
00332 }
00333
00334 void pausarJuego()
00335 {
00336   juego.jugando = false;
00337   pthread_join(ttiempo, NULL);
00338   pthread_join(tautos, NULL);
00339   pthread_join(tjugador, NULL);
00340 }
00341
00342 void actualizarInterfaz()
00343 {
00344   actualizarMapa();
00345   copiarMatriz(disp_matriz, juego.mapa);
00346   disp_matriz[0] = juego.vidas;
00347   clock_t frac = juego.tiempo_inicio » 4, aux = juego.tiempo;
00348   disp_matriz[1] = 0;
00349   while (aux > 0)
00350   {
00351     disp_matriz[1] «= 1;
00352     disp_matriz[1] |= 1;
00353     aux -= frac;
00354   }
00355
00356   disp_matriz[(juego.jugador_posicion_sur) - 1] |= juego.jugador_1;
00357   disp_matriz[juego.jugador_posicion_sur] |= juego.jugador_2;
00358
00359   actualizarDisplay();
00360
00361   if ((juego.mapa[(juego.jugador_posicion_sur) - 1]) & juego.jugador_1 ||
     (juego.mapa[(juego.jugador_posicion_sur)]) & juego.jugador_2 || juego.timeout)
00362     perderVida();
00363 }
00364
00365 void reanudarJuego(void)
00366 {
00367   juego.jugando = true;
00368   pthread_create(&ttiempo, NULL, threadTiempo, NULL);
00369   pthread_create(&tjugador, NULL, threadJugador, NULL);
00370   pthread_create(&tautos, NULL, threadAutos, NULL);
00371 }
00372
00373 /*****************************************************************************
00374  *****************************************************************************
00375                           LOCAL FUNCTION DEFINITIONS
00376  *****************************************************************************
00377  *****************************************************************************/
00378
00379 static void *threadTiempo(void *ptr)
00380 {
00381   clock_t ref = clock();
00382   while (juego.jugando)
00383   {
00384     if (!(juego.timeout))
00385     {
00386       juego.tiempo = juego.tiempo_referencia - (clock() - ref);
00387       juego.timeout = juego.tiempo <= 0;
00388     }
00389     else
00390       ref = clock();
00391   }
00392
00393   juego.tiempo_referencia = juego.tiempo;
00394
00395   return NULL;
00396 }
00397
00398 static void *threadAutos(void *ptr)
00399 {
00400   clock_t ref = clock();
00401   while (juego.jugando)
00402   {
00403     if (!juego.timeout && !juego.refresco_autos)
00404       juego.refresco_autos = (clock() - ref) > juego.tiempo_refresco_autos;
00405     else
```

```
00406       ref = clock();
00407   }
00408   return NULL;
00409 }
00410
00411 static void *threadJugador(void *ptr)
00412 {
00413   clock_t ref = clock();
00414   while (juego.jugando)
00415   {
00416     if (!juego.timeout && !juego.refresco_jugador)
00417       juego.refresco_jugador = (clock() - ref) > juego.tiempo_refresco_jugador;
00418     else
00419       ref = clock();
00420   }
00421   return NULL;
00422 }
```

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

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

Include dependency graph for game_data.c:



Data Structures

- struct data_t

Macros

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

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_LIFE** , **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_time_left_flag (void)

  Avisa si se excedio el tiempo de juego.
- 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.

4.38.1 Detailed Description

Author

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ∼ Ingeniería Electrónica ∼ ITBA

Definition in file game_data.c.

4.38.2 Macro Definition Documentation

4.38.2.1 EXTRA_TIME_PER_BONUS_GOAL `#define EXTRA_TIME_PER_BONUS_GOAL 15`

Definition at line 35 of file game_data.c.

4.38.2.2 EXTRA_TIME_PER_GOAL `#define EXTRA_TIME_PER_GOAL 10`

Definition at line 34 of file game_data.c.

4.38.2.3 INITIAL_RUN_TIME_LEFT `#define INITIAL_RUN_TIME_LEFT 30`

Definition at line 32 of file game_data.c.

4.38.2.4 MAX_LIVES `#define MAX_LIVES 3`

Definition at line 26 of file game_data.c.

4.38.2.5 MAX_NAME_CHAR `#define MAX_NAME_CHAR 20`

Definition at line 24 of file game_data.c.

4.38.2.6 SCORE_PER_GOAL `#define SCORE_PER_GOAL 500`

Definition at line 28 of file game_data.c.

4.38.2.7 SCORE_PER_GOAL_COIN `#define SCORE_PER_GOAL_COIN 750`

Definition at line 29 of file game_data.c.

4.38.2.8 SCORE_PER_RUN `#define SCORE_PER_RUN 1000`

Definition at line 30 of file game_data.c.

4.38.2.9 TIME_LEFT_WARNING `#define TIME_LEFT_WARNING 10`

Definition at line 37 of file game_data.c.

4.38.3 Enumeration Type Documentation

4.38.3.1 DATA_FLAGS `enum DATA_FLAGS`

Definition at line 70 of file game_data.c.

4.38.3.2 HUD_EXTRAS `enum HUD_EXTRAS`

Definition at line 78 of file game_data.c.

4.38.4 Function Documentation

4.38.4.1 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 332 of file game_data.c.

4.38.4.2 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 290 of file game_data.c.

4.38.4.3 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 296 of file game_data.c.

4.38.4.4 game_data_add_score() `void game_data_add_score ( void )`

Agrega score por llegar a la meta.

Definition at line 253 of file game_data.c.

4.38.4.5 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 259 of file game_data.c.

4.38.4.6 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 391 of file game_data.c.

4.38.4.7 game_data_clear_name() `void game_data_clear_name (`

`void )`

Limpia el nombre del jugador.

Definition at line 322 of file game_data.c.

4.38.4.8 game_data_draw() `void game_data_draw (`

`void )`

Grafica datos del juego (HUD pricipalmente)

Definition at line 232 of file game_data.c.

4.38.4.9 game_data_get_diff() `int game_data_get_diff (`

`void )`

Devuelve dificultad.

Returns

int

Definition at line 317 of file game_data.c.

4.38.4.10 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 302 of file game_data.c.

4.38.4.11 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 383 of file game_data.c.

4.38.4.12 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 354 of file game_data.c.

4.38.4.13 game_data_get_lives() `int game_data_get_lives ( void )`

Devuelve vidas.

Returns

int vidas

Definition at line 238 of file game_data.c.

4.38.4.14 game_data_get_name() `char * game_data_get_name ( void )`

Devuelve puntero al nombre del jugador.

Returns

char*

Definition at line 349 of file game_data.c.

4.38.4.15 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 409 of file game_data.c.

4.38.4.16 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 275 of file game_data.c.

4.38.4.17 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 285 of file game_data.c.

4.38.4.18 game_data_get_score() `unsigned long long game_data_get_score ( void )`

Devuelve score.

Returns

int

Definition at line 248 of file game_data.c.

4.38.4.19 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 270 of file game_data.c.

4.38.4.20 game_data_get_time_left_flag() bool game_data_get_time_left_flag (void)

Avisa si se excedio el tiempo de juego.

Returns

true Excedido

false No excedido

Definition at line 371 of file game_data.c.

4.38.4.21 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 307 of file game_data.c.

4.38.4.22 game_data_init() void game_data_init (void)

Inicializa datos internos del juego.

Definition at line 170 of file game_data.c.

4.38.4.23 game_data_next_run() void game_data_next_run (void)

Indica que se pase a la siguiente run.

Definition at line 280 of file game_data.c.

4.38.4.24 game_data_overwrite_name() void game_data_overwrite_name (char * *name*)

Sobreescribe el nombre del jugador.

Parameters

name	

Definition at line 327 of file game_data.c.

4.38.4.25 game_data_reset_goals() `void game_data_reset_goals ( void )`

Habilita todos los goals.

Definition at line 364 of file game_data.c.

4.38.4.26 game_data_set_diff() `void game_data_set_diff ( int diff )`

Setea dificultad.

Parameters

diff	enum DIFFICULTIES

Definition at line 312 of file game_data.c.

4.38.4.27 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 359 of file game_data.c.

4.38.4.28 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 265 of file game_data.c.

4.38.4.29 game_data_subtract_live() `void game_data_subtract_live ( void )`

Resta una vida.

Definition at line 243 of file game_data.c.

4.38.4.30 game_data_update() `void game_data_update ( void )`

Actualiza datos internos del juego.

Definition at line 188 of file game_data.c.

4.38.5 Variable Documentation

4.38.5.1 flag `bool flag`

Definition at line 88 of file game_data.c.

4.38.5.2 timer `int timer`

Definition at line 90 of file game_data.c.

4.38.5.3 value `int value`

Definition at line 89 of file game_data.c.

4.39 game_data.c

Go to the documentation of this file.

```
00001
00010 /*******************************************************************************
00011  * INCLUDE HEADER FILES
00012  ******************************************************************************/
00013
00014 #include "game_data.h"
00015 #include <stdio.h>
00016 #include <time.h>
00017 #include "geometry.h"
00018 #include "allegro_stuff.h"
00019
00020 /*******************************************************************************
00021  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00022  ******************************************************************************/
00023
00024 #define MAX_NAME_CHAR 20
00025
00026 #define MAX_LIVES 3
00027
00028 #define SCORE_PER_GOAL 500      // puntaje por llegar a la meta
00029 #define SCORE_PER_GOAL_COIN 750 // puntaje por llegar a la meta con coin
00030 #define SCORE_PER_RUN 1000      // puntaje por completar una run
00031
00032 #define INITIAL_RUN_TIME_LEFT 30
00033
00034 #define EXTRA_TIME_PER_GOAL 10       // 10s extras por llegar a una meta
00035 #define EXTRA_TIME_PER_BONUS_GOAL 15 // 15s extras por llegar a una meta con coin
00036
00037 #define TIME_LEFT_WARNING 10 // warning 10s antes del timeout
00038
00039 /*******************************************************************************
00040  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00041  ******************************************************************************/
00042
00043 typedef struct
00044 {
00045   int lives;
00046   unsigned long long score;
00047   unsigned long long score_max;
00048
00049   struct
00050   {
00051     int number;    // numero de run actual
00052     int time_left; // tiempo restante en la run
00053     int time;      // tiempo de la run actual
00054     long time_ref; // referencia de tiempo global de la run
00055   } run;
00056
00057   unsigned long frames;
00058   int timer_in_sec;
00059
00060   int difficulty;
00061
00062   char name[MAX_NAME_CHAR];
00063
00064   unsigned char flag;
00065
00066   bool goals[MAX_GOALS];
00067
00068 } data_t;
00069
00070 enum DATA_FLAGS
00071 {
00072   DATA_FLAG_STARTING,
00073   DATA_FLAG_NEXT_RUN,
00074   DATA_FLAG_TIME_EXCEEDED,
00075   DATA_FLAG_GAME_OVER
00076 };
00077
00078 enum HUD_EXTRAS
00079 {
00080   HUD_EXTRA_TIME,
00081   HUD_EXTRA_SCORE,
00082   HUD_EXTRA_LIFE,
00083   HUD_EXTRAS_MAX
00084 };
00085
00086 static struct
00087 {
00088   bool flag;
00089   int value;
00090   int timer;
00091 } hud_extra_stuff[HUD_EXTRAS_MAX];
```

```
00092
00093 /*****************************************************************************
00094  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00095  *****************************************************************************/
00096
00101 static void data_init(void);
00102
00107 static void data_update(void);
00108
00113 static void hud_draw(void);
00114
00119 static void draw_reached_goals(void);
00120
00125 static void next_run(void);
00126
00132 static void trigger_show_adding_time(int extra);
00133
00134 /*****************************************************************************
00135  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00136  *****************************************************************************/
00137
00138 // Datos internos del juego
00139 static data_t data;
00140
00141 // Referencia temporal del inicio del juego
00142 static long time_ref;
00143
00144 static int char_h; // altura de un caracter
00145 static int char_w; // ancho de un caracter
00146
00147 static ALLEGRO_COLOR text_color;
00148
00149 // Flag para triggear sonido de poco tiempo
00150 static bool flag_low_time_warning;
00151
00152 // Auxiliar para hacer acciones en los cambios de segundos
00153 static int last_loop_time;
00154
00155 // Tiempo restante inicial de una nueva run. Se puede modificar externamente
00156 static int new_run_time_left;
00157
00158 // Auxiliar para mostrar el score gradualmente en el HUD
00159 static unsigned long long score_display;
00160
00161 // Score maximo no actualizado en game over
00162 static unsigned long long max_score_no_updated;
00163
00164 /*****************************************************************************
00165  *****************************************************************************
00166             GLOBAL FUNCTION DEFINITIONS
00167  *****************************************************************************
00168  *****************************************************************************/
00169
00170 void game_data_init(void)
00171 {
00172   time_ref = time(NULL);
00173
00174   char_h = allegro_get_var_font_h();
00175   char_w = allegro_get_var_font_w();
00176
00177   text_color = al_map_rgb(255, 255, 255);
00178
00179   flag_low_time_warning = false;
00180
00181   new_run_time_left = INITIAL_RUN_TIME_LEFT;
00182
00183   score_display = 0;
00184
00185   data_init();
00186 }
00187
00188 void game_data_update(void)
00189 {
00190   data_update();
00191
00192   if (data.flag == DATA_FLAG_NEXT_RUN)
00193   {
00194     next_run();
00195     data.flag = DATA_FLAG_STARTING;
00196   }
00197
00198   if (data.run.time_left == TIME_LEFT_WARNING && !flag_low_time_warning)
00199   {
00200     allegro_sound_play_effect_low_time();
00201
00202     flag_low_time_warning = true;
00203   }
```

```
00204   else if (data.run.time_left < TIME_LEFT_WARNING)
00205     flag_low_time_warning = false;
00206
00207   if (max_score_no_updated != data.score_max)
00208     max_score_no_updated = data.score_max;
00209
00210   if (data.run.time_left == 0)
00211   {
00212     // data.flag = DATA_FLAG_TIME_EXCEEDED;
00213     game_data_subtract_live();
00214     allegro_sound_play_effect_no_time();
00215     data.flag = DATA_FLAG_GAME_OVER;
00216   }
00217
00218   if (data.lives == 0)
00219   {
00220     data.flag = DATA_FLAG_GAME_OVER;
00221   }
00222
00223   // Las primeras 2 runs se reduce el timer en 5 segundos.
00224   if (data.run.number <= 1)
00225     new_run_time_left = INITIAL_RUN_TIME_LEFT - (5 * (data.run.number + 1));
00226
00227   // Despues se reduce de a 2 segundos por run hasta llegar a 10 segundos.
00228   else if (data.run.number <= 6)
00229     new_run_time_left = INITIAL_RUN_TIME_LEFT - (2 * (data.run.number + 1));
00230 }
00231
00232 void game_data_draw(void)
00233 {
00234   hud_draw();
00235   draw_reached_goals();
00236 }
00237
00238 int game_data_get_lives(void)
00239 {
00240   return (data.lives);
00241 }
00242
00243 void game_data_subtract_live(void)
00244 {
00245   data.lives--;
00246 }
00247
00248 unsigned long long game_data_get_score(void)
00249 {
00250   return (data.score);
00251 }
00252
00253 void game_data_add_score(void)
00254 {
00255
00256   data.score += SCORE_PER_GOAL;
00257 }
00258
00259 void game_data_add_score_bonus(void)
00260 {
00261
00262   data.score += SCORE_PER_GOAL_COIN;
00263 }
00264
00265 void game_data_set_score_max(unsigned long long score)
00266 {
00267   data.score_max = score;
00268 }
00269
00270 unsigned long long game_data_get_score_max(void)
00271 {
00272   return data.score_max;
00273 }
00274
00275 int game_data_get_run_number(void)
00276 {
00277   return (data.run.number);
00278 }
00279
00280 void game_data_next_run(void)
00281 {
00282   data.flag = DATA_FLAG_NEXT_RUN;
00283 }
00284
00285 int game_data_get_run_time_left(void)
00286 {
00287   return (data.run.time_left);
00288 }
00289
00290 void game_data_add_run_time_goal(void)
```

```
00291 {
00292   data.run.time_left += EXTRA_TIME_PER_GOAL;
00293   trigger_show_adding_time(EXTRA_TIME_PER_GOAL);
00294 }
00295
00296 void game_data_add_run_time_goal_bonus(void)
00297 {
00298   data.run.time_left += EXTRA_TIME_PER_BONUS_GOAL;
00299   trigger_show_adding_time(EXTRA_TIME_PER_BONUS_GOAL);
00300 }
00301
00302 unsigned long game_data_get_frames(void)
00303 {
00304   return (data.frames);
00305 }
00306
00307 int game_data_get_timer_in_sec(void)
00308 {
00309   return (data.timer_in_sec);
00310 }
00311
00312 void game_data_set_diff(int diff)
00313 {
00314   data.difficulty = diff;
00315 }
00316
00317 int game_data_get_diff(void)
00318 {
00319   return (data.difficulty);
00320 }
00321
00322 void game_data_clear_name(void)
00323 {
00324   memset(data.name, 0, MAX_NAME_CHAR);
00325 }
00326
00327 void game_data_overwrite_name(char *name)
00328 {
00329   strcpy(data.name, name);
00330 }
00331
00332 void game_data_add_name_letter(char letter)
00333 {
00334   int length = strlen(data.name);
00335
00336   if ((letter == ALLEGRO_KEY_BACKSPACE) && (length > 0))
00337   {
00338     data.name[length - 1] = 0;
00339   }
00340
00341   else if (letter >= ALLEGRO_KEY_A && letter <= ALLEGRO_KEY_Z && length < MAX_NAME_CHAR)
00342   {
00343     letter += '@';
00344     data.name[length] = letter;
00345     data.name[length + 1] = 0;
00346   }
00347 }
00348
00349 char *game_data_get_name(void)
00350 {
00351   return (data.name);
00352 }
00353
00354 bool game_data_get_goal_state(unsigned int goal)
00355 {
00356   return data.goals[goal];
00357 }
00358
00359 void game_data_set_goal(unsigned int goal)
00360 {
00361   data.goals[goal] = true;
00362 }
00363
00364 void game_data_reset_goals(void)
00365 {
00366   int i;
00367   for (i = 0; i < MAX_GOALS; i++)
00368     data.goals[i] = false;
00369 }
00370
00371 bool game_data_get_time_left_flag(void)
00372 {
00373   if (data.flag == DATA_FLAG_TIME_EXCEEDED)
00374   {
00375     data.flag = DATA_FLAG_STARTING;
00376     return true;
00377   }
```

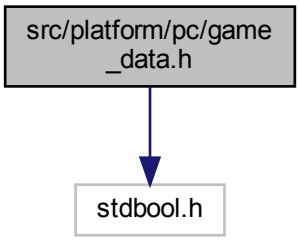
```
00378
00379   else
00380     return false;
00381 }
00382
00383 bool game_data_get_game_over_flag(void)
00384 {
00385   if (data.flag == DATA_FLAG_GAME_OVER)
00386     return true;
00387   else
00388     return false;
00389 }
00390
00391 bool game_data_are_goals_full(void)
00392 {
00393   bool state = true;
00394
00395   int i;
00396   for (i = 0; i < MAX_GOALS; i++)
00397   {
00398     // si alguno esa vacio...
00399     if (!data.goals[i])
00400     {
00401       state = false;
00402       break;
00403     }
00404   }
00405
00406   return state;
00407 }
00408
00409 unsigned long long game_data_get_old_max_score(void)
00410 {
00411   return max_score_no_updated;
00412 }
00413
00414 /*****************************************************************************
00415  *****************************************************************************
00416             LOCAL FUNCTION DEFINITIONS
00417  *****************************************************************************
00418  *****************************************************************************/
00419
00420 static void data_init(void)
00421 {
00422   data.frames = 0;
00423   data.lives = MAX_LIVES;
00424   data.run.number = 0;
00425   data.run.time_left = new_run_time_left;
00426   data.run.time = 0;
00427   data.run.time_ref = time(NULL);
00428   data.score = 0;
00429   data.timer_in_sec = 0;
00430
00431   data.flag = DATA_FLAG_STARTING;
00432
00433   last_loop_time = 0;
00434
00435   game_data_reset_goals();
00436 }
00437
00438 static void data_update(void)
00439 {
00440
00441   // diferencia entre el tiempo actual y el de referencia
00442   data.timer_in_sec = time(NULL) - time_ref;
00443   data.frames++;
00444
00445   data.run.time = time(NULL) - data.run.time_ref;
00446
00447   if (data.run.time > last_loop_time)
00448   {
00449     data.run.time_left--;
00450     last_loop_time++;
00451   }
00452   last_loop_time = data.run.time;
00453 }
00454
00455 static void hud_draw(void)
00456 {
00457   // Dibuja la puntuacion en pantalla.
00458
00459   int i;
00460
00461   // Graduacion del score a mostrar para que vaya incrementando de apoco
00462   if (score_display != data.score)
00463   {
00464     int shifter;
```

```
00465
00466     for (i = 2, shifter = 0; i > 0; i--)
00467     {
00468       shifter = 1 « i;
00469       if (score_display <= (data.score - shifter))
00470         score_display += shifter;
00471     }
00472   }
00473
00474   al_draw_textf(
00475       allegro_get_var_font(),
00476       text_color, // Negro porque por ahora sigue el fondo blanco, sino recomiendo amarillo (255, 255,
      51).
00477       1, 1,       // Arriba a la izquierda.
00478       0,
00479       "Score: %06lld", // 6 cifras (por ahi es mucho).
00480       score_display);
00481
00482   // Dibuja el numero de vuelta.
00483   al_draw_textf(
00484       allegro_get_var_font(),
00485       text_color,
00486       1, CELL_H - char_h - 5, // Para que quede abaj de la puntuacion en pantalla.
00487       0,
00488       "Run: %02d", // 2 cifras. No me acuerdo si esta bien asi.
00489       data.run.number);
00490
00491   // Segundos.
00492   al_draw_textf(
00493       allegro_get_var_font(),
00494       text_color,
00495       al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w, 1,
00496       0,
00497       "Played Time: %04d",
00498       data.timer_in_sec);
00499
00500   // Tiempo restante
00501   al_draw_textf(
00502       allegro_get_var_font(),
00503       text_color,
00504       al_get_text_width(allegro_get_var_font(), "Score: xxxxxx") + 3 * char_w,
00505       CELL_H - char_h - 5,
00506       0,
00507       "Time Left: %03d",
00508       data.run.time_left);
00509
00510   if (hud_extra_stuff[HUD_EXTRA_TIME].flag)
00511   {
00512   }
00513
00514   // Dibuja vidas.
00515   for (int i = 0; i < data.lives; i++) // No se si la rana tiene 'frog.lives' pero aca va el
      equivalente.
00516     al_draw_bitmap(
00517         sprites.heart,
00518         // DISPLAY_W - SPRITE_SIZE_HEART * (data.lives - i), 1,         //Arriba a la derecha.
      'LIFE_W' depende de la imagen que usemos.
00519         DISPLAY_W - 100 + SPRITE_SIZE_HEART * (data.lives - i - 1),
00520         (CELL_H - char_h - 5) / 2,
00521         0);
00522
00523   /*
00524   if(!data.lives)
00525     al_draw_text(
00526       allegro_get_var_font(),
00527       al_map_rgb(255, 255, 51),       //Amarillo, es el color que mas se ditingue en general.
00528       DISPLAY_W / 2, DISPLAY_H / 2,
00529       ALLEGRO_ALIGN_CENTER,           //Para que se dibuje en el medio.
00530       "G A M E  O V E R");
00531   */
00532 }
00533
00534 static void draw_reached_goals(void)
00535 {
00536   int i;
00537   for (i = 0; i < MAX_GOALS; i++)
00538   {
00539     // si algun goal fue alcanzado...
00540     if (data.goals[i])
00541       al_draw_bitmap(sprites.frog[6],
00542                      goal_cols[i] * CELL_W + FROG_OFFSET_X - 1,
00543                      CELL_H + FROG_OFFSET_Y + GOAL_ROW_OFFSET_Y_FIX,
00544                      0);
00545   }
00546 }
00547
00548 static void next_run(void)
```

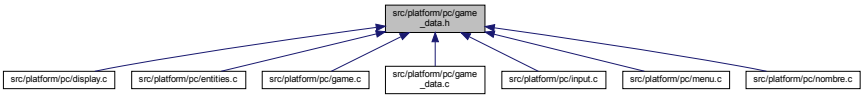
```
00549 {
00550   data.run.number++;
00551   data.run.time_left = new_run_time_left;
00552   data.run.time = 0;
00553   data.run.time_ref = time(NULL);
00554
00555   data.score += SCORE_PER_RUN;
00556
00557   last_loop_time = 0;
00558
00559   flag_low_time_warning = false;
00560
00561   game_data_reset_goals();
00562 }
00563
00564 static void trigger_show_adding_time(int extra)
00565 {
00566 }
```

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

`#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_time_left_flag (void)

  Avisa si se excedio el tiempo de juego.

- 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.

4.40.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-17

Copyright

Copyright (c) 2022

Definition in file game_data.h.

4.40.2 Enumeration Type Documentation

4.40.2.1 DIFFICULTIES `enum DIFFICULTIES`

Definition at line 25 of file game_data.h.

4.40.3 Function Documentation

4.40.3.1 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 332 of file game_data.c.

4.40.3.2 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 290 of file game_data.c.

4.40.3.3 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 296 of file game_data.c.

4.40.3.4 game_data_add_score() `void game_data_add_score ( void )`

Agrega score por llegar a la meta.

Definition at line 253 of file game_data.c.

4.40.3.5 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 259 of file game_data.c.

4.40.3.6 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 391 of file game_data.c.

4.40.3.7 game_data_clear_name() `void game_data_clear_name (`

`void )`

Limpia el nombre del jugador.

Definition at line 322 of file game_data.c.

4.40.3.8 game_data_draw() `void game_data_draw (`

`void )`

Grafica datos del juego (HUD pricipalmente)

Definition at line 232 of file game_data.c.

4.40.3.9 game_data_get_diff() `int game_data_get_diff (`

`void )`

Devuelve dificultad.

Returns

int

Definition at line 317 of file game_data.c.

4.40.3.10 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 302 of file game_data.c.

4.40.3.11 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 383 of file game_data.c.

4.40.3.12 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 354 of file game_data.c.

4.40.3.13 game_data_get_lives() `int game_data_get_lives ( void )`

Devuelve vidas.

Returns

int vidas

Definition at line 238 of file game_data.c.

4.40.3.14 game_data_get_name() `char * game_data_get_name ( void )`

Devuelve puntero al nombre del jugador.

Returns

char*

Definition at line 349 of file game_data.c.

4.40.3.15 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 409 of file game_data.c.

4.40.3.16 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 275 of file game_data.c.

4.40.3.17 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 285 of file game_data.c.

4.40.3.18 game_data_get_score() unsigned long long game_data_get_score (void)

Devuelve score.

Returns

int

Definition at line 248 of file game_data.c.

4.40.3.19 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 270 of file game_data.c.

4.40.3.20 game_data_get_time_left_flag() bool game_data_get_time_left_flag (void)

Avisa si se excedio el tiempo de juego.

Returns

true Excedido

false No excedido

Definition at line 371 of file game_data.c.

4.40.3.21 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 307 of file game_data.c.

4.40.3.22 game_data_init() void game_data_init (void)

Inicializa datos internos del juego.

Definition at line 170 of file game_data.c.

4.40.3.23 game_data_next_run() void game_data_next_run (void)

Indica que se pase a la siguiente run.

Definition at line 280 of file game_data.c.

4.40.3.24 game_data_overwrite_name() void game_data_overwrite_name (char * *name*)

Sobreescribe el nombre del jugador.

Parameters

name	

Definition at line 327 of file game_data.c.

4.40.3.25 game_data_reset_goals() `void game_data_reset_goals ( void )`

Habilita todos los goals.

Definition at line 364 of file game_data.c.

4.40.3.26 game_data_set_diff() `void game_data_set_diff ( int diff )`

Setea dificultad.

Parameters

diff	enum DIFFICULTIES

Definition at line 312 of file game_data.c.

4.40.3.27 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 359 of file game_data.c.

4.40.3.28 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 265 of file game_data.c.

4.40.3.29 game_data_subtract_live() void game_data_subtract_live (void)

Resta una vida.

Definition at line 243 of file game_data.c.

4.40.3.30 game_data_update() void game_data_update (void)

Actualiza datos internos del juego.

Definition at line 188 of file game_data.c.

4.41 game_data.h

Go to the documentation of this file.

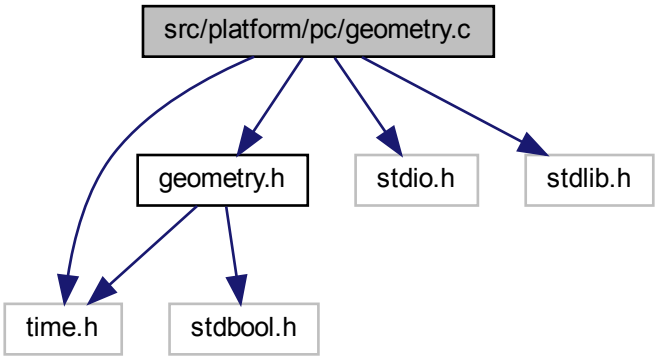
```
00001
00012 #ifndef _GAME_DATA_H_
00013 #define _GAME_DATA_H_
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include <stdbool.h>
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_time_left_flag(void);
00215
00222 bool game_data_get_game_over_flag(void);
00223
00230 bool game_data_are_goals_full(void);
00231
00237 unsigned long long game_data_get_old_max_score(void);
00238
00239 /*****************************************************************************
00240  *****************************************************************************/
00241
00242 #endif // _GAME_DATA_H_
```

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

```
#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.
- bool matchUint (unsigned int val, const unsigned int *array)

  Verifica si un numero coincide con alguno de un array dado (uints)

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.

4.42.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-10

Copyright

Copyright (c) 2022

Definition in file geometry.c.

4.42.2 Function Documentation

4.42.2.1 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.

4.42.2.2 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 133 of file geometry.c.

4.42.2.3 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.

4.42.2.4 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 186 of file geometry.c.

4.42.2.5 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 191 of file geometry.c.

4.42.2.6 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 176 of file geometry.c.

4.42.2.7 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 196 of file geometry.c.

4.42.2.8 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 181 of file geometry.c.

4.42.2.9 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 138 of file geometry.c.

4.42.2.10 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 150 of file geometry.c.

4.42.2.11 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 155 of file geometry.c.

4.42.2.12 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 165 of file geometry.c.

4.42.2.13 matchUint()

```
bool matchUint (
    unsigned int val,
    const unsigned int * array )
```

Verifica si un numero coincide con alguno de un array dado (uints)

Parameters

val	Numero a verificar
array	Array de numeros

Returns

true Existe

false No existe

Definition at line 201 of file geometry.c.

4.42.3 Variable Documentation

4.42.3.1 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.

4.42.3.2 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.

4.42.3.3 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.

4.42.3.4 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.

4.43 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
```

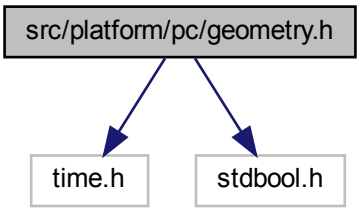
```
00110 /*****************************************************************************
00111  *****************************************************************************
00112             GLOBAL FUNCTION DEFINITIONS
00113  *****************************************************************************
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   if (ax1 < bx2 &&
00124       ax2 > bx1 &&
00125       ay1 < by2 &&
00126       ay2 > by1)
00127
00128     return true;
00129
00130   return false;
00131 }
00132
00133 bool collideShort(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)
00134 {
00135   return (collide(ax, ay, ax + aw, ay + ah, bx, by, bx + bw, by + bh));
00136 }
00137
00138 bool inside(int ax1, int ay1, int ax2, int ay2, int bx1, int by1, int bx2, int by2)
00139 {
00140   if (bx1 > ax1 &&
00141       by1 > ay1 &&
00142       bx2 < ax2 &&
00143       by2 < ay2)
00144
00145     return true;
00146
00147   return false;
00148 }
00149
00150 bool insideShort(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh)
00151 {
00152   return (inside(ax, ay, ax + aw, ay + ah, bx, by, bx + bw, by + bh));
00153 }
00154
00155 bool insideShortScaled(int ax, int ay, int aw, int ah, int bx, int by, int bw, int bh, float scale)
00156 {
00157   if (scale < 0.0 || scale > 1.0)
00158     return false;
00159
00160   float diff = (bw * (1.0 - scale));
00161
00162   return (inside(ax, ay, ax + aw, ay + ah, bx + diff, by, bx + bw - diff, by + bh));
00163 }
00164
00165 int mapInt(int source, int min_in, int max_in, int min_out, int max_out)
00166 {
00167   // int slope = (max_out - max_in) / (min_out - min_in);
00168
00169   // int output = min_out + (slope * (source - min_in));
00170
00171   int output = (source - min_in) * (max_out - min_out) / (max_in - min_in) + min_out;
00172
00173   return (output);
00174 }
00175
00176 pair_xy_t getXYFromFrogFrame(int frame)
00177 {
00178   return (pair_xy_frog_sprites_frames[frame]);
00179 }
00180
00181 pair_xy_t getXYFromTurtleFrame(int frame)
00182 {
00183   return (pair_xy_turtle_sprites_frames[frame]);
00184 }
00185
00186 pair_xy_t getXYFromCarFrame(int frame)
00187 {
00188   return (pair_xy_car_sprites_frames[frame]);
00189 }
00190
00191 pair_xy_t getXYFromCoinFrame(int frame)
00192 {
00193   return (pair_xy_coin_sprites_frames[frame]);
00194 }
00195
00196 pair_xy_t getXYFromSplashFrame(int frame)
```

```
00197 {
00198   return (pair_xy_splash_sprites_frames[frame]);
00199 }
00200
00201 bool matchUint(unsigned int val, const unsigned int *array)
00202 {
00203   if (array == NULL)
00204   {
00205     printf("PUNTERO INVALIDO ~ funcion: matchUint ~ val=%d", val);
00206     exit(EXIT_FAILURE);
00207   }
00208
00209   int i;
00210   for (i = 0; array[i] != '\0'; i++)
00211   {
00212     if (val == array[i])
00213       return true;
00214     ;
00215   }
00216
00217   return false;
00218 }
```

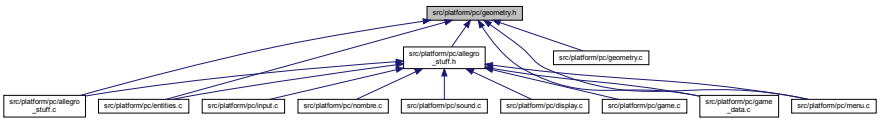
4.44 src/platform/pc/geometry.h File Reference

```
#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 20
- #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.
- bool matchUint (unsigned int val, const unsigned int *array)

  Verifica si un numero coincide con alguno de un array dado (uints)

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.

4.44.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-10

Copyright

Definition in file geometry.h.

4.44.2 Macro Definition Documentation

4.44.2.1 CAR_H `#define CAR_H 40`

Definition at line 87 of file geometry.h.

4.44.2.2 CAR_OFFSET_X `#define CAR_OFFSET_X 0`

Definition at line 88 of file geometry.h.

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

Definition at line 89 of file geometry.h.

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

Definition at line 85 of file geometry.h.

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

Definition at line 86 of file geometry.h.

4.44.2.6 CAR_W `#define CAR_W CELL_W + 26`

Definition at line 84 of file geometry.h.

4.44.2.7 CELL_H `#define CELL_H 46`

Definition at line 30 of file geometry.h.

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

Definition at line 56 of file geometry.h.

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

Definition at line 57 of file geometry.h.

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

Definition at line 46 of file geometry.h.

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

Definition at line 47 of file geometry.h.

4.44.2.12 CELL_TOPLEFT_X `#define CELL_TOPLEFT_X 0`

Definition at line 42 of file geometry.h.

4.44.2.13 CELL_TOPLEFT_Y `#define CELL_TOPLEFT_Y 0`

Definition at line 43 of file geometry.h.

4.44.2.14 CELL_W `#define CELL_W 46`

Definition at line 31 of file geometry.h.

4.44.2.15 COLS `#define COLS 15`

Definition at line 29 of file geometry.h.

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

Definition at line 123 of file geometry.h.

4.44.2.17 CREDITS_SCREEN_LENGTH `#define CREDITS_SCREEN_LENGTH 2576`

Definition at line 121 of file geometry.h.

4.44.2.18 CREDITS_SCREEN_START `#define CREDITS_SCREEN_START 0`

Definition at line 122 of file geometry.h.

4.44.2.19 DISPLAY_H `#define DISPLAY_H 644`

Definition at line 27 of file geometry.h.

4.44.2.20 DISPLAY_W `#define DISPLAY_W 690`

Definition at line 26 of file geometry.h.

4.44.2.21 FROG_FRAMES `#define FROG_FRAMES 8`

Definition at line 60 of file geometry.h.

4.44.2.22 FROG_H `#define FROG_H 30`

Definition at line 50 of file geometry.h.

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

Definition at line 72 of file geometry.h.

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

Definition at line 73 of file geometry.h.

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

Definition at line 74 of file geometry.h.

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

Definition at line 75 of file geometry.h.

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

Definition at line 52 of file geometry.h.

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

Definition at line 53 of file geometry.h.

4.44.2.29 FROG_W `#define FROG_W 30`

Definition at line 49 of file geometry.h.

4.44.2.30 GOAL_ROW_MARGIN_TO_REACH `#define GOAL_ROW_MARGIN_TO_REACH 5`

Definition at line 129 of file geometry.h.

4.44.2.31 GOAL_ROW_OFFSET_Y_FIX `#define GOAL_ROW_OFFSET_Y_FIX 5`

Definition at line 128 of file geometry.h.

4.44.2.32 INSERTION_FACTOR `#define INSERTION_FACTOR (double)0.5`

Definition at line 126 of file geometry.h.

4.44.2.33 LANES_CAR_TOTAL `#define LANES_CAR_TOTAL 5`

Definition at line 35 of file geometry.h.

4.44.2.34 LANES_LOG_TOTAL `#define LANES_LOG_TOTAL 3`

Definition at line 37 of file geometry.h.

4.44.2.35 LANES_TURTLE_TOTAL `#define LANES_TURTLE_TOTAL 2`

Definition at line 39 of file geometry.h.

4.44.2.36 LOG_H `#define LOG_H 40`

Definition at line 79 of file geometry.h.

4.44.2.37 LOG_OFFSET_X `#define LOG_OFFSET_X 0`

Definition at line 80 of file geometry.h.

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

Definition at line 81 of file geometry.h.

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

Definition at line 78 of file geometry.h.

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

Definition at line 33 of file geometry.h.

4.44.2.41 MENU_OPTION_DELTA_Y `#define MENU_OPTION_DELTA_Y 100`

Definition at line 117 of file geometry.h.

4.44.2.42 MENU_OPTION_H `#define MENU_OPTION_H 75`

Definition at line 119 of file geometry.h.

4.44.2.43 MENU_OPTION_TOPLEFT_X `#define MENU_OPTION_TOPLEFT_X 45`

Definition at line 115 of file geometry.h.

4.44.2.44 MENU_OPTION_TOPLEFT_Y `#define MENU_OPTION_TOPLEFT_Y 72`

Definition at line 116 of file geometry.h.

4.44.2.45 MENU_OPTION_W `#define MENU_OPTION_W 600`

Definition at line 118 of file geometry.h.

4.44.2.46 ROWS `#define ROWS 14`

Definition at line 28 of file geometry.h.

4.44.2.47 SPRITE_BORDER_START_X `#define SPRITE_BORDER_START_X 0`

Definition at line 112 of file geometry.h.

4.44.2.48 SPRITE_BORDER_START_Y `#define SPRITE_BORDER_START_Y CELL_H`

Definition at line 113 of file geometry.h.

4.44.2.49 SPRITE_COIN_FRAMES `#define SPRITE_COIN_FRAMES 6`

Definition at line 102 of file geometry.h.

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

Definition at line 104 of file geometry.h.

4.44.2.51 SPRITE_COIN_SIDE `#define SPRITE_COIN_SIDE 24`

Definition at line 103 of file geometry.h.

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

Definition at line 100 of file geometry.h.

4.44.2.53 SPRITE_DEAD_SIZE `#define SPRITE_DEAD_SIZE 35`

Definition at line 99 of file geometry.h.

4.44.2.54 SPRITE_SIZE_FROG_DYNAMIC_LONG `#define SPRITE_SIZE_FROG_DYNAMIC_LONG 46`

Definition at line 63 of file geometry.h.

4.44.2.55 SPRITE_SIZE_FROG_DYNAMIC_SHORT `#define SPRITE_SIZE_FROG_DYNAMIC_SHORT FROG_W`

Definition at line 64 of file geometry.h.

4.44.2.56 SPRITE_SIZE_FROG_STATIC_H `#define SPRITE_SIZE_FROG_STATIC_H FROG_H`

Definition at line 61 of file geometry.h.

4.44.2.57 SPRITE_SIZE_FROG_STATIC_W `#define SPRITE_SIZE_FROG_STATIC_W FROG_W`

Definition at line 62 of file geometry.h.

4.44.2.58 SPRITE_SIZE_HEART `#define SPRITE_SIZE_HEART 20`

Definition at line 97 of file geometry.h.

4.44.2.59 SPRITE_SPLASH_FRAMES `#define SPRITE_SPLASH_FRAMES 6`

Definition at line 106 of file geometry.h.

4.44.2.60 SPRITE_SPLASH_H `#define SPRITE_SPLASH_H 68`

Definition at line 108 of file geometry.h.

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

Definition at line 109 of file geometry.h.

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

Definition at line 110 of file geometry.h.

4.44.2.63 SPRITE_SPLASH_W `#define SPRITE_SPLASH_W 98`

Definition at line 107 of file geometry.h.

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

Definition at line 69 of file geometry.h.

4.44.2.65 STEP_FULL_SIZE `#define STEP_FULL_SIZE CELL_H`

Definition at line 67 of file geometry.h.

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

Definition at line 68 of file geometry.h.

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

Definition at line 94 of file geometry.h.

4.44.2.68 TURTLE_FRAMES `#define TURTLE_FRAMES 11`

Definition at line 92 of file geometry.h.

4.44.2.69 TURTLE_SIDE `#define TURTLE_SIDE CELL_W`

Definition at line 93 of file geometry.h.

4.44.3 Enumeration Type Documentation

4.44.3.1 CAR_TYPE `enum CAR_TYPE`

Definition at line 180 of file geometry.h.

4.44.3.2 DIRECTIONS `enum DIRECTIONS`

Definition at line 152 of file geometry.h.

4.44.3.3 GOALS enum GOALS

Definition at line 142 of file geometry.h.

4.44.3.4 MENU_STATES enum MENU_STATES

Definition at line 161 of file geometry.h.

4.44.3.5 MENU_WINDOWS enum MENU_WINDOWS

Definition at line 171 of file geometry.h.

4.44.4 Function Documentation

4.44.4.1 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.

4.44.4.2 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 133 of file geometry.c.

4.44.4.3 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.

4.44.4.4 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 186 of file geometry.c.

4.44.4.5 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 191 of file geometry.c.

4.44.4.6 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 176 of file geometry.c.

4.44.4.7 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 196 of file geometry.c.

4.44.4.8 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 181 of file geometry.c.

4.44.4.9 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 138 of file geometry.c.

4.44.4.10 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 150 of file geometry.c.

4.44.4.11 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 155 of file geometry.c.

4.44.4.12 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 165 of file geometry.c.

4.44.4.13 matchUint()

```
bool matchUint (
    unsigned int val,
    const unsigned int * array )
```

Verifica si un numero coincide con alguno de un array dado (uints)

Parameters

val	Numero a verificar
array	Array de numeros

Returns

true Existe

false No existe

Definition at line 201 of file geometry.c.

4.44.5 Variable Documentation

4.44.5.1 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.

4.44.5.2 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.

4.44.5.3 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.

4.44.5.4 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.

4.45 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 20 // 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
00374 bool matchUint(unsigned int val, const unsigned int *array);
00375
00376 /*****************************************************************************
00377  *****************************************************************************/
00378
00379 #endif // _GEOMETRY_H_
```

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

```
#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.

4.46.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-22

Copyright

Copyright (c) 2022

Definition in file input.c.

4.46.2 Function Documentation

4.46.2.1 iniciarEntradas() void iniciarEntradas (void)

Inicializa las entradas de la plataforma.

Definition at line 36 of file input.c.

4.46.2.2 leerEntradas() event_t leerEntradas (void)

Devuelve una entrada válida.

Returns

event_t enum eventos_tecla

Definition at line 40 of file input.c.

4.47 input.c

Go to the documentation of this file.

```
00001
00012 /*****************************************************************************
00013  * INCLUDE HEADER FILES
00014  *****************************************************************************/
00015
00016 #include "../../input.h"
00017
00018 #include "allegro_stuff.h"
00019 #include "entities.h"
00020 #include "game_data.h"
00021
00022 /*****************************************************************************
00023  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00024  *****************************************************************************/
00025
00026 static ALLEGRO_EVENT *event;
00027
00028 static unsigned char last_key;
00029
00030 /*****************************************************************************
00031  *****************************************************************************
00032                          GLOBAL FUNCTION DEFINITIONS
00033  *****************************************************************************
00034  *****************************************************************************/
00035
00036 void iniciarEntradas(void)
00037 {
00038 }
00039
00040 event_t leerEntradas(void)
00041 {
00042   event_t retorno = NO_MOVER;
00043   // bool queue_no_empty;
00044
00045   event = allegro_get_next_event();
00046   int i;
00047
00048   if (event != NULL)
00049   {
00050     switch ((*event).type)
00051     {
00052     case ALLEGRO_EVENT_TIMER:
00053       allegro_set_var_redraw(true);
00054
00055       break;
00056
00057     case ALLEGRO_EVENT_KEY_DOWN:
00058       last_key = allegro_get_last_key();
```

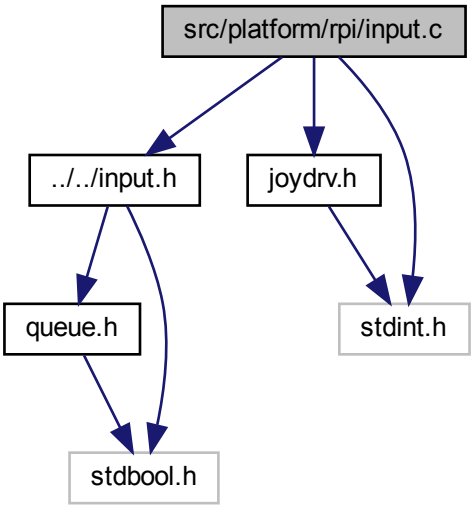
```
00059
00060       if (last_key != (*event).keyboard.keycode)
00061       {
00062         retorno = (*event).keyboard.keycode;
00063         allegro_set_last_key(retorno);
00064
00065         switch (retorno)
00066         {
00067         case ALLEGRO_KEY_F2:
00068           allegro_sound_set_stream_gain_up();
00069           break;
00070
00071         case ALLEGRO_KEY_F1:
00072           allegro_sound_set_stream_gain_down();
00073           break;
00074
00075         case ALLEGRO_KEY_8:
00076           break;
00077
00078         case ALLEGRO_KEY_9:
00079           break;
00080
00081         case ALLEGRO_KEY_1:
00082           game_data_add_score();
00083           break;
00084
00085         case ALLEGRO_KEY_2:
00086           retorno = GAME_OVER;
00087           break;
00088
00089         case ALLEGRO_KEY_3:
00090           // int i;
00091           for (i = 0; i < MAX_GOALS; i++)
00092             game_data_set_goal(i);
00093           break;
00094
00095         case ALLEGRO_KEY_4:
00096           game_data_reset_goals();
00097           break;
00098
00099         default:
00100           break;
00101         }
00102       }
00103
00104       break;
00105
00106     case ALLEGRO_EVENT_KEY_UP:
00107       allegro_set_last_key(0);
00108
00109       break;
00110
00111     case ALLEGRO_EVENT_DISPLAY_CLOSE:
00112       retorno = FORCE_SALIR;
00113       break;
00114
00115     default:
00116       break;
00117     }
00118   }
00119
00120   return retorno;
00121 }
```

4.48 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.

4.48.1 Detailed Description

Archivo para manejo del joystick en RPI.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Definition in file input.c.

4.48.2 Function Documentation

4.48.2.1 iniciarEntradas() void iniciarEntradas (void)

Inicializa las entradas de la plataforma.

Definition at line 41 of file input.c.

4.48.2.2 leerEntradas() event_t leerEntradas (void)

Devuelve una entrada válida.

Returns

event_t enum eventos_tecla

Definition at line 46 of file input.c.

4.49 input.c

Go to the documentation of this file.

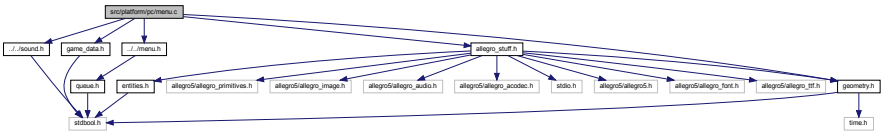
```
00001
00008 /*******************************************************************************
00009  * INCLUDE HEADER FILES
00010  ******************************************************************************/
00011
00012 #include "../../input.h"
00013 #include "joydrv.h"
00014
00015 #include <stdint.h>
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 }
```

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

```
#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)

4.50.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-22

Copyright

Copyright (c) 2022

Definition in file menu.c.

4.50.2 Macro Definition Documentation

4.50.2.1 STATS_X_COORD `#define STATS_X_COORD 20`

Definition at line 27 of file menu.c.

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

Definition at line 28 of file menu.c.

4.50.3 Function Documentation

4.50.3.1 bajarOpcion() `void bajarOpcion (`

```
void )
```

Selecciona la opcion inferior a la actual.

Definition at line 154 of file menu.c.

4.50.3.2 destruirMenu() `void destruirMenu (`

```
void )
```

Destruye del menu.

Definition at line 87 of file menu.c.

4.50.3.3 getOpcion() `int getOpcion (`

```
void )
```

Devuelve la opcion actual del menu.

Returns

int Opcion seleccionada actualmente

Definition at line 134 of file menu.c.

4.50.3.4 iniciarMenu() `void iniciarMenu (`

```
void )
```

Inicia el menu.

Definition at line 82 of file menu.c.

4.50.3.5 moverOpcionActual() `void moverOpcionActual (`

```
void )
```

Definition at line 169 of file menu.c.

4.50.3.6 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 92 of file menu.c.

4.50.3.7 setOpcion() `void setOpcion ( int opc )`

Selecciona una opcion del menu.

Parameters

opc	Opcion a seleccionar

Definition at line 126 of file menu.c.

4.50.3.8 subirOpcion() `void subirOpcion ( void )`

Selecciona la opcion superior a la actual.

Definition at line 139 of file menu.c.

4.51 menu.c

Go to the documentation of this file.

```
00001
00012 /*****************************************************************************
00013  * INCLUDE HEADER FILES
00014  *****************************************************************************/
00015
00016 #include "../../menu.h"
00017 #include "../../sound.h"
00018
00019 #include "allegro_stuff.h"
00020 #include "geometry.h"
00021 #include "game_data.h"
00022
00023 /*****************************************************************************
00024  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00025  *****************************************************************************/
00026
00027 #define STATS_X_COORD 20
00028 #define STATS_Y_COORD_START (DISPLAY_H / 2 + 50)
00029
00030 /*****************************************************************************
00031  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00032  *****************************************************************************/
00033
00034 typedef struct
00035 {
00036   int actual_state;
00037   int max_states;
00038 } window_t;
00039
```

```
00040 typedef struct
00041 {
00042   window_t window[MENU_WINDOW_MAX];
00043
00044   int actual_window;
00045
00046 } menu_t;
00047
00048 /*****************************************************************************
00049  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00050  *****************************************************************************/
00051
00056 static void inicializarMenu(void);
00057
00062 static void renderizarMenu(void);
00063
00068 static void show_stats(void);
00069
00070 /*****************************************************************************
00071  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00072  *****************************************************************************/
00073
00074 static menu_t menu;
00075
00076 /*****************************************************************************
00077  *****************************************************************************
00078                         GLOBAL FUNCTION DEFINITIONS
00079  *****************************************************************************
00080  *****************************************************************************/
00081
00082 void iniciarMenu(void)
00083 {
00084   inicializarMenu();
00085 }
00086
00087 void destruirMenu(void)
00088 {
00089   allegro_deinits();
00090 }
00091
00092 void setMenu(int *a, unsigned int size)
00093 {
00094   switch (a[0])
00095   {
00096   // menu principal (JUGAR, DIFICULTAD, RANKING, SALIRTXT)
00097   case JUGAR:
00098     menu.actual_window = MENU_WINDOW_HOME;
00099
00100     break;
00101
00102   // menu dificultades (FACIL, NORMAL, DIFICIL)
00103   case FACIL:
00104     menu.actual_window = MENU_WINDOW_DIFFICULTY;
00105
00106     break;
00107
00108   // menu pausa (CONTINUAR, REINICIAR, SALIRTXT)
00109   case CONTINUAR:
00110     menu.actual_window = MENU_WINDOW_PAUSE;
00111     allegro_set_rick_flag(false);
00112
00113     break;
00114
00115   // menu game over (REINICIAR, SALIRTXT)
00116   case REINICIAR:
00117     menu.actual_window = MENU_WINDOW_GAME_OVER;
00118
00119     break;
00120
00121   default:
00122     break;
00123   }
00124 }
00125
00126 void setOpcion(int opc)
00127 {
00128   // Seleccina uno de los botones del menu actual
00129   menu.window[menu.actual_window].actual_state = opc;
00130
00131   renderizarMenu();
00132 }
00133
00134 int getOpcion(void)
00135 {
00136   return (menu.window[menu.actual_window].actual_state);
00137 }
00138
```

```
00139 void subirOpcion(void)
00140 {
00141   int *actual_option = &menu.window[menu.actual_window].actual_state;
00142   int *max_option = &menu.window[menu.actual_window].max_states;
00143 
00144   (*actual_option)--;
00145 
00146   if (*actual_option < 0)
00147     setOpcion(*max_option - 1);
00148   else
00149     renderizarMenu();
00150 
00151   reproducirEfecto(EFECTO_SELECCION);
00152 }
00153 
00154 void bajarOpcion(void)
00155 {
00156   int *actual_option = &menu.window[menu.actual_window].actual_state;
00157   int *max_option = &menu.window[menu.actual_window].max_states;
00158 
00159   (*actual_option)++;
00160 
00161   if (*actual_option == *max_option)
00162     setOpcion(0);
00163   else
00164     renderizarMenu();
00165 
00166   reproducirEfecto(EFECTO_SELECCION);
00167 }
00168 
00169 void moverOpcionActual(void)
00170 {
00171 }
00172 
00173 /*****************************************************************************
00174  *****************************************************************************
00175                          LOCAL FUNCTION DEFINITIONS
00176  *****************************************************************************
00177  *****************************************************************************/
00178 
00179 static void inicializarMenu(void)
00180 {
00181   // menu principal (JUGAR, DIFICULTAD, RANKING, SALIRTXT)
00182   menu.window[MENU_WINDOW_HOME].max_states = 5;
00183 
00184   // menu dificultades (FACIL, NORMAL, DIFICIL)
00185   menu.window[MENU_WINDOW_DIFFICULTY].max_states = 3;
00186 
00187   // menu pausa (CONTINUAR, REINICIAR, SALIRTXT)
00188   menu.window[MENU_WINDOW_PAUSE].max_states = 3;
00189 
00190   // menu game over (REINICIAR, SALIRTXT)
00191   menu.window[MENU_WINDOW_GAME_OVER].max_states = 2;
00192 }
00193 
00194 static void renderizarMenu()
00195 {
00196   allegro_clear_display();
00197 
00198   ALLEGRO_BITMAP *background = NULL;
00199   ALLEGRO_BITMAP *option = NULL;
00200 
00201   background = sprites.menu[menu.actual_window].background;
00202   option = sprites.menu[menu.actual_window].option[menu.window[menu.actual_window].actual_state];
00203 
00204   al_draw_bitmap(background, 0, 0, 0);
00205 
00206   al_draw_bitmap(option, MENU_OPTION_TOPLEFT_X,
00207                  MENU_OPTION_TOPLEFT_Y + (menu.window[menu.actual_window].actual_state *
       MENU_OPTION_DELTA_Y),
00208                  0);
00209 
00210   if (menu.actual_window == MENU_WINDOW_PAUSE || menu.actual_window == MENU_WINDOW_GAME_OVER)
00211     show_stats();
00212 
00213   al_flip_display();
00214 }
00215 
00216 static void show_stats(void)
00217 {
00218   char *name = game_data_get_name();
00219   int score = game_data_get_score();
00220   int max_score = game_data_get_old_max_score();
00221   ALLEGRO_COLOR color = al_map_rgb(255, 255, 255);
00222   ALLEGRO_FONT *font = allegro_get_var_font();
00223 
00224   int x = STATS_X_COORD;
```

```
00225   int y = STATS_Y_COORD_START;
00226
00227   al_draw_textf(font, color,
00228                 x,
00229                 y,
00230                 0,
00231                 "Jugador: %s", name);
00232
00233   y += 20;
00234
00235   if (menu.actual_window == MENU_WINDOW_PAUSE)
00236   {
00237     al_draw_textf(font, color,
00238                   x,
00239                   y,
00240                   0,
00241                   "Score:   %d", score);
00242
00243     y += 20;
00244
00245     al_draw_textf(font, color,
00246                   x,
00247                   y,
00248                   0,
00249                   "Max score:   %d", max_score);
00250   }
00251   else if (menu.actual_window == MENU_WINDOW_GAME_OVER)
00252   {
00253     if (score > max_score)
00254     {
00255       al_draw_textf(font, color,
00256                     x,
00257                     y + 10,
00258                     0,
00259                     "NUEVA PUNTUACION MAXIMA!");
00260
00261       y += 40;
00262
00263       al_draw_textf(font, color,
00264                     x,
00265                     y,
00266                     0,
00267                     "Anterior: %d", max_score);
00268
00269       y += 20;
00270
00271       al_draw_textf(font, color,
00272                     x,
00273                     y,
00274                     0,
00275                     "Nueva: %d", score);
00276     }
00277
00278     else
00279     {
00280       al_draw_textf(font, color,
00281                     x,
00282                     y,
00283                     0,
00284                     "Score:   %d", score);
00285
00286       y += 20;
00287
00288       al_draw_textf(font, color,
00289                     x,
00290                     y,
00291                     0,
00292                     "Max score:   %d", max_score);
00293     }
00294   }
00295 }
```

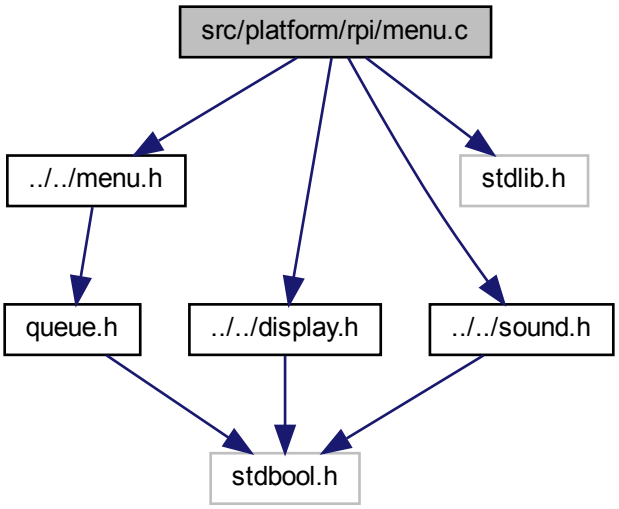
4.52 src/platform/rpi/menu.c File Reference

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.

4.52.1 Detailed Description

Archivo para manejo de los menús en RPI.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file menu.c.

4.52.2 Function Documentation

4.52.2.1 bajarOpcion() `void bajarOpcion (`

```
void  )
```

Selecciona la opcion inferior a la actual.

Definition at line 72 of file menu.c.

4.52.2.2 destruirMenu() `void destruirMenu (`

```
void  )
```

Destruye del menu.

Definition at line 85 of file menu.c.

4.52.2.3 getOpcion() `int getOpcion (`

```
void  )
```

Devuelve la opcion actual del menu.

Returns

int Opcion seleccionada actualmente

Definition at line 58 of file menu.c.

4.52.2.4 iniciarMenu() `void iniciarMenu (`

```
void  )
```

Inicia el menu.

Definition at line 81 of file menu.c.

4.52.2.5 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.

4.52.2.6 setOpcion() `void setOpcion ( int opc )`

Selecciona una opcion del menu.

Parameters

opc	Opcion a seleccionar

Definition at line 52 of file menu.c.

4.52.2.7 subirOpcion() `void subirOpcion ( void )`

Selecciona la opcion superior a la actual.

Definition at line 63 of file menu.c.

4.52.3 Variable Documentation

4.52.3.1 max_opciones `int max_opciones`

Definition at line 26 of file menu.c.

4.52.3.2 menu_actual `int* menu_actual`

Definition at line 24 of file menu.c.

4.52.3.3 opcion_actual `int opcion_actual`

Definition at line 25 of file menu.c.

4.53 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   menu.menu_actual = realloc(menu.menu_actual, size * sizeof(int));
00044   int i;
00045   for (i = 0; i < size; i++)
00046   {
00047     menu.menu_actual[i] = a[i];
00048   }
00049   menu.max_opciones = size;
00050 }
00051
00052 void setOpcion(int opc)
00053 {
00054   menu.opcion_actual = opc;
00055   dejarTexto(menu_textos[menu.menu_actual[menu.opcion_actual]], POS_OPCION, true);
00056 }
00057
00058 int getOpcion()
00059 {
00060   return menu.opcion_actual;
00061 }
00062
00063 void subirOpcion()
00064 {
00065   if (--menu.opcion_actual < 0)
00066     menu.opcion_actual = menu.max_opciones - 1;
00067   dejarTexto(menu_textos[menu.menu_actual[menu.opcion_actual]], POS_OPCION, true);
00068
00069   reproducirEfecto(EFECTO_SELECCION);
00070 }
00071
00072 void bajarOpcion()
00073 {
00074   if (++menu.opcion_actual >= menu.max_opciones)
00075     menu.opcion_actual = 0;
00076   dejarTexto(menu_textos[menu.menu_actual[menu.opcion_actual]], POS_OPCION, true);
00077
00078   reproducirEfecto(EFECTO_SELECCION);
00079 }
00080
00081 void iniciarMenu()
00082 {
00083 }
00084
00085 void destruirMenu()
00086 {
00087   free(menu.menu_actual);
00088 }
```

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

```
#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.

4.54.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-22

Copyright

Copyright (c) 2022

Definition in file nombre.c.

4.54.2 Macro Definition Documentation

4.54.2.1 NAME_TOPLEFT_X `#define NAME_TOPLEFT_X 55`

Definition at line 27 of file nombre.c.

4.54.2.2 NAME_TOPLEFT_Y `#define NAME_TOPLEFT_Y 312`

Definition at line 28 of file nombre.c.

4.54.3 Function Documentation

4.54.3.1 agregarLetra() `void agregarLetra ( void )`

Confirma la letra.

Definition at line 61 of file nombre.c.

4.54.3.2 bajarLetra() `void bajarLetra ( void )`

Selecciona la letra inferior.

Definition at line 53 of file nombre.c.

4.54.3.3 devolverNombre() `char * devolverNombre ( void )`

Devuelve puntero al nombre.

Returns

char* Puntero al nombre

Definition at line 82 of file nombre.c.

4.54.3.4 nuevoNombre() `void nuevoNombre (`

`void )`

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

Definition at line 36 of file nombre.c.

4.54.3.5 siguienteLetra() `void siguienteLetra (`

`void )`

Confirma la letra y pasa a seleccionar la siguiente.

Definition at line 57 of file nombre.c.

4.54.3.6 subirLetra() `void subirLetra (`

`void )`

Selecciona la siguiente letra superior.

Definition at line 49 of file nombre.c.

4.54.3.7 subirNombre() `void subirNombre (`

`void )`

Definition at line 78 of file nombre.c.

4.55 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 {
```

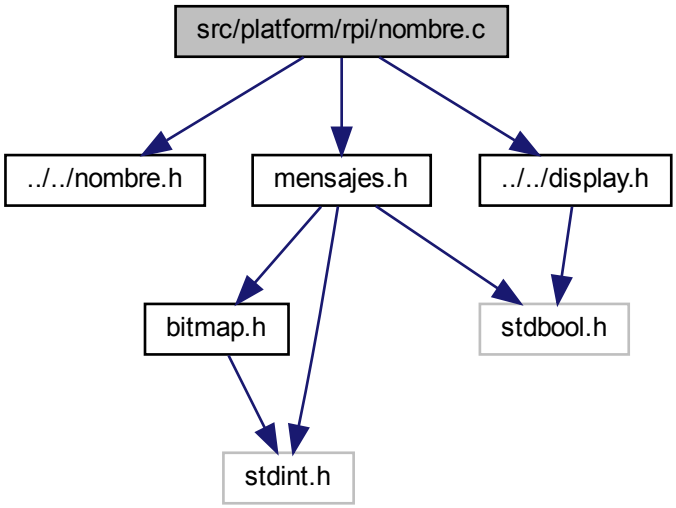
```
00038   allegro_clear_display();
00039
00040   game_data_clear_name();
00041   game_data_set_score_max(0);
00042
00043   /*cambiar por background correspondiente*/
00044   al_draw_bitmap(sprites.name, 0, 0, 0);
00045
00046   al_flip_display();
00047 }
00048
00049 void subirLetra(void)
00050 {
00051 }
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 }
```

4.56 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

4.56.1 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.

4.56.2 Function Documentation

4.56.2.1 agregarLetra() `void agregarLetra ( void )`

Confirma la letra.

Definition at line 71 of file nombre.c.

4.56.2.2 bajarLetra() `void bajarLetra ( void )`

Selecciona la letra inferior.

Definition at line 53 of file nombre.c.

4.56.2.3 devolverNombre() `char * devolverNombre ( void )`

Devuelve puntero al nombre.

Returns

char* Puntero al nombre

Definition at line 75 of file nombre.c.

4.56.2.4 nuevoNombre() `void nuevoNombre ( void )`

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

Definition at line 35 of file nombre.c.

4.56.2.5 siguienteLetra() `void siguienteLetra ( void )`

Confirma la letra y pasa a seleccionar la siguiente.

Definition at line 62 of file nombre.c.

4.56.2.6 subirLetra() void subirLetra (void)

Selecciona la siguiente letra superior.

Definition at line 44 of file nombre.c.

4.56.3 Variable Documentation

4.56.3.1 disp_matriz matriz_t disp_matriz [extern]

Definition at line 38 of file display.c.

4.57 nombre.c

Go to the documentation of this file.

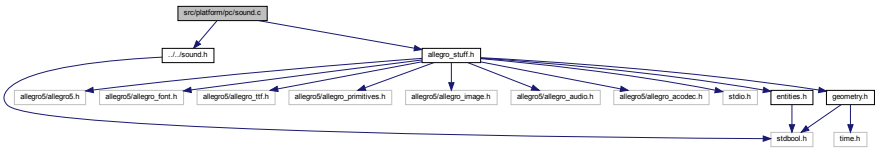
```
00001
00008 /*********************************************************************************
00009  * INCLUDE HEADER FILES
00010  *********************************************************************************/
00011
00012 #include "../../nombre.h"
00013 #include "mensajes.h"
00014 #include "../../display.h"
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   last = 'A';
00039   printRenglon(nombre.renglon);
00040   copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00041   actualizarDisplay();
00042 }
00043
00044 void subirLetra()
00045 {
00046   if (--last < 'A')
00047     last = 'Z';
00048   reemplazarUltLetraMensaje(last, &nombre);
00049   copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00050   actualizarDisplay();
00051 }
00052
00053 void bajarLetra()
00054 {
00055   if (++last > 'Z')
00056     last = 'A';
00057   reemplazarUltLetraMensaje(last, &nombre);
00058   copiarMatrizRenglon(disp_matriz, nombre.renglon, POS_MSJ2);
00059   actualizarDisplay();
```

```
00060 }
00061
00062 void siguienteLetra()
00063 {
00064   concatenarLetraMensaje(last, &nombre);
00065
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 }
```

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

```
#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.

4.58.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

2022-01-27

Copyright

Copyright (c) 2022

Definition in file sound.c.

4.58.2 Function Documentation

4.58.2.1 destruirSonido() `void destruirSonido ( void )`

Desinicializa el sonido de la plataforma.

Definition at line 31 of file sound.c.

4.58.2.2 iniciarSonido() `bool iniciarSonido ( void )`

Inicializa el sonido de la plataforma.

Returns

true Exito

false Error

Definition at line 26 of file sound.c.

4.58.2.3 pausarMusica() `void pausarMusica ( void )`

Pausa la musica actual.

Definition at line 35 of file sound.c.

4.58.2.4 reproducirEfecto() `void reproducirEfecto ( int efecto )`

Pone a reproducir un efecto dado.

Parameters

int	num efectos

Definition at line 75 of file sound.c.

4.58.2.5 reproducirMusica() `void reproducirMusica ( int musica )`

Pone a reproducir una musica dada.

Parameters

int	enum musica

Definition at line 40 of file sound.c.

4.59 sound.c

Go to the documentation of this file.

```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "../../sound.h"
00017
00018 #include "allegro_stuff.h"
00019
00020 /*******************************************************************************
00021  *******************************************************************************
00022                          GLOBAL FUNCTION DEFINITIONS
00023  *******************************************************************************
00024  ******************************************************************************/
00025
00026 bool iniciarSonido(void)
00027 {
00028   return true;
00029 }
00030
00031 void destruirSonido(void)
00032 {
00033 }
00034
00035 void pausarMusica(void)
00036 {
00037   allegro_sound_pause_stream();
00038 }
00039
00040 void reproducirMusica(int musica)
00041 {
00042   switch (musica)
00043   {
00044   case MUSICA_CREDITOS:
00045     allegro_sound_set_stream_credits();
00046     break;
00047
00048   case MUSICA_JUGANDO:
00049     allegro_sound_set_stream_playing();
00050     break;
00051
00052   case MUSICA_MENU_PAUSA:
00053     allegro_sound_set_stream_pause_menu();
00054     break;
00055
00056   case MUSICA_MENU_PPAL:
```

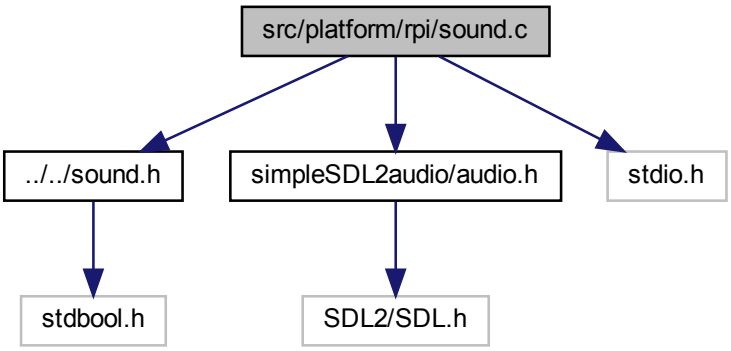
```
00057     allegro_sound_set_stream_main_menu();
00058     break;
00059
00060   case MUSICA_RANKING:
00061     allegro_sound_set_stream_ranking();
00062     break;
00063
00064   case MUSICA_GAME_OVER:
00065     allegro_sound_set_stream_game_over();
00066     break;
00067
00068   default:
00069     break;
00070   }
00071
00072   allegro_sound_play_stream();
00073 }
00074
00075 void reproducirEfecto(int sonido)
00076 {
00077   switch (sonido)
00078   {
00079   case EFECTO_AHOGADO:
00080     allegro_sound_play_effect_drowned();
00081     break;
00082
00083   case EFECTO_IMPACTO:
00084     allegro_sound_play_effect_crash();
00085     break;
00086
00087   case EFECTO_MENU_ENTER:
00088     allegro_sound_play_effect_menu_enter();
00089     break;
00090
00091   case EFECTO_META:
00092     allegro_sound_play_effect_goal();
00093     break;
00094
00095   case EFECTO_NIVEL_COMPLETO:
00096     allegro_sound_play_effect_run_completed();
00097     break;
00098
00099   case EFECTO_NUEVO_MAX_SCORE:
00100     allegro_sound_play_effect_new_max_score();
00101     break;
00102
00103   case EFECTO_POCO_TIEMPO:
00104     allegro_sound_play_effect_low_time();
00105     break;
00106
00107   case EFECTO_SALIENDO:
00108     allegro_sound_play_effect_exiting();
00109     break;
00110
00111   case EFECTO_SALTO:
00112     allegro_sound_play_effect_jump();
00113     break;
00114
00115   case EFECTO_SELECCION:
00116     allegro_sound_play_effect_click();
00117     break;
00118
00119   default:
00120     break;
00121   }
00122 }
```

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

Archivo para manejo del sonido en RPI.

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

4.60.1 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.

4.60.2 Macro Definition Documentation

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

Definition at line 22 of file sound.c.

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

Definition at line 21 of file sound.c.

4.60.3 Function Documentation

4.60.3.1 destruirSonido() `void destruirSonido ( void )`

Desinicializa el sonido de la plataforma.

Definition at line 77 of file sound.c.

4.60.3.2 iniciarSonido() `bool iniciarSonido ( void )`

Inicializa el sonido de la plataforma.

Returns

true Exito
false Error

Definition at line 57 of file sound.c.

4.60.3.3 pausarMusica() `void pausarMusica ( void )`

Pausa la musica actual.

Definition at line 84 of file sound.c.

4.60.3.4 reproducirEfecto() `void reproducirEfecto ( int efecto )`

Pone a reproducir un efecto dado.

Parameters

int	num efectos

Definition at line 98 of file sound.c.

4.60.3.5 reproducirMusica() `void reproducirMusica ( int musica )`

Pone a reproducir una musica dada.

Parameters

int	enum musica

Definition at line 89 of file sound.c.

4.61 sound.c

Go to the documentation of this file.

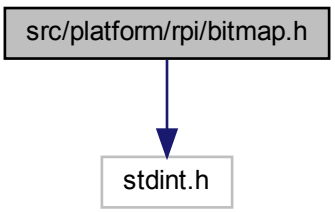
```
00001
00008 /*******************************************************************************
00009  * INCLUDE HEADER FILES
00010  ******************************************************************************/
00011
00012 #include "../../sound.h"
00013 #include "simpleSDL2audio/audio.h"
00014
00015 #include <stdio.h>
00016
00017 /*******************************************************************************
00018  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00019  ******************************************************************************/
00020
00021 #define MUSICA_DIR "../res/sounds/streams"
00022 #define EFECTOS_DIR "../res/sounds/samples"
00023
00024 /*******************************************************************************
00025  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00026  ******************************************************************************/
00027
00028 static char *archivos_musica[] =
00029     {MUSICA_DIR "/main_menu_theme.wav",
00030      MUSICA_DIR "/ranking_theme.wav",
00031      MUSICA_DIR "/credits_theme.wav",
00032      MUSICA_DIR "/playing_theme.wav",
00033      MUSICA_DIR "/pause_menu_theme.wav",
00034      MUSICA_DIR "/game_over.wav"};
00035
00036 static char *archivos_efectos[] =
00037     {EFECTOS_DIR "/click.wav",
00038      EFECTOS_DIR "/jump_original.wav",
00039      EFECTOS_DIR "/crash.wav",
00040      EFECTOS_DIR "/fall_in_water.wav",
00041      EFECTOS_DIR "/low_time.wav",
00042      EFECTOS_DIR "/goal_reached.wav",
00043      EFECTOS_DIR "/run_completed.wav",
00044      EFECTOS_DIR "/new_max_score.wav",
00045      EFECTOS_DIR "/menu_enter.wav",
00046      EFECTOS_DIR "/saliendo.wav",
00047      EFECTOS_DIR "/no_time.wav"};
00048
00049 static Audio *musica;
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
00066     /*
00067   if (initAudio() == NO_INIT)
00068   {
00069     printf("Audio not initilized.\n");
00070     endAudio();
00071     return false;
00072   }
00073     */
00074   return true;
00075 }
00076
00077 void destruirSonido(void)
00078 {
00079   endAudio();
00080   freeAudio(musica);
00081     SDL_Quit();
00082 }
00083
00084 void pausarMusica(void)
00085 {
00086   pauseAudio();
00087 }
00088
00089 void reproducirMusica(int m)
00090 {
00091   pauseAudio();
00092   freeAudio(musica);
00093   musica = createAudio(archivos_musica[m], 1, SDL_MIX_MAXVOLUME);
00094   unpauseAudio();
00095   playMusicFromMemory(musica, SDL_MIX_MAXVOLUME);
00096 }
00097
00098 void reproducirEfecto(int e)
00099 {
00100   playSound(archivos_efectos[e], SDL_MIX_MAXVOLUME);
00101 }
```

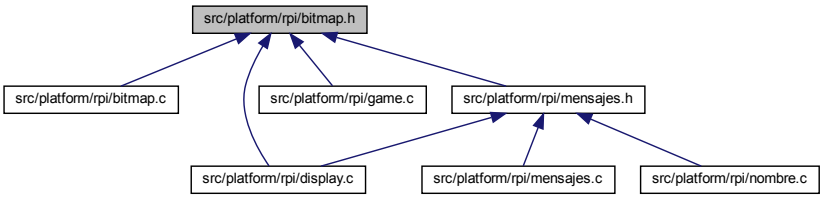
4.62 src/platform/rpi/bitmap.h File Reference

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".

4.62.1 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.

4.62.2 Macro Definition Documentation

4.62.2.1 CANT_COLUMNAS `#define CANT_COLUMNAS 16`

Definition at line 22 of file bitmap.h.

4.62.2.2 CANT_FILAS `#define CANT_FILAS 16`

Definition at line 21 of file bitmap.h.

4.62.3 Typedef Documentation

4.62.3.1 matriz_t `typedef uint16_t matriz_t[CANT_FILAS]`

Definition at line 28 of file bitmap.h.

4.62.4 Function Documentation

4.62.4.1 copiarMatriz()

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

Copia el contenido de una matriz en otra.

Parameters

destino	
desde	

Definition at line 37 of file bitmap.c.

4.62.4.2 limpiarMatriz()

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

Borra el contenido de una matriz.

Parameters

A	

Definition at line 31 of file bitmap.c.

4.62.4.3 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 43 of file bitmap.c.

4.62.4.4 matrizNot()

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

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

Parameters

A	

Definition at line 61 of file bitmap.c.

4.62.4.5 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 49 of file bitmap.c.

4.62.4.6 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 55 of file bitmap.c.

4.62.4.7 printMatriz() `void printMatriz (`

```
matriz_t A )
```

Imprime una matriz en consola (para debug)

Parameters

A	

Definition at line 22 of file bitmap.c.

4.63 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_
```

4.64 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
```

4.65 joydrv.h

```
00001
00012 #ifndef JOYDRV_H
00013 #define JOYDRV_H
00014
00015 /*****************************************************************************
00016  * INCLUDE HEADER FILES
00017  *****************************************************************************/
00018
00019 #include <stdint.h>
```

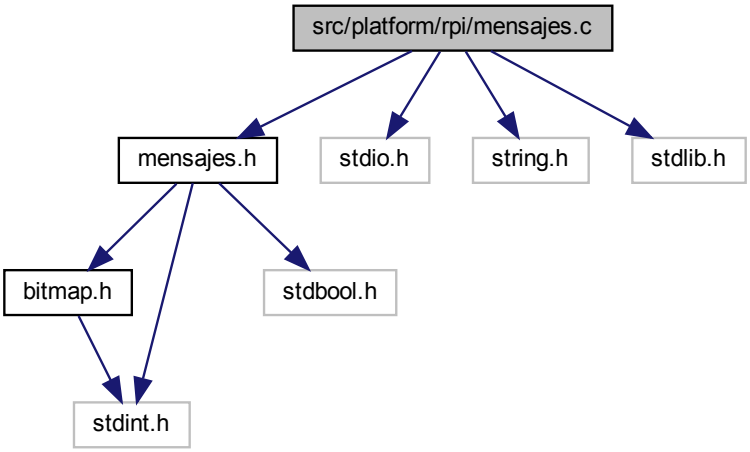
```
00020
00021 /*****************************************************************************
00022  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00023  *****************************************************************************/
00024
00025 // Las coordenadas del joystick varían entre -128 y 127 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
```

4.66 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

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

4.66.1 Detailed Description

Permite codificar strings en formato renglon para mostrar en display.

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file mensajes.c.

4.66.2 Macro Definition Documentation

4.66.2.1 CANT_SIMBOLOS `#define CANT_SIMBOLOS 38`

Definition at line 25 of file mensajes.c.

4.66.2.2 INDEX_CERO `#define INDEX_CERO 27`

Definition at line 23 of file mensajes.c.

4.66.2.3 INDEX_ESPACIO `#define INDEX_ESPACIO 26`

Definition at line 22 of file mensajes.c.

4.66.2.4 INDEX_FULL `#define INDEX_FULL 37`

Definition at line 24 of file mensajes.c.

4.66.2.5 PEDIR_FULL `#define PEDIR_FULL -1`

Definition at line 27 of file mensajes.c.

4.66.3 Function Documentation

4.66.3.1 borrarRenglon() `void borrarRenglon ( renglon_t r )`

elimina el contenido del Renglon

Parameters

r	Renglon a limpiar

Definition at line 118 of file mensajes.c.

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

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

Parameters

c	caracter
m	Matriz
x	
y	

Definition at line 196 of file mensajes.c.

4.66.3.3 charARenglon() `void charARenglon (`

```
char c,
renglon_t r )
```

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

Parameters

c	caracter
r	Renglon

Definition at line 176 of file mensajes.c.

4.66.3.4 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 312 of file mensajes.c.

4.66.3.5 copiarMatrizRenglon() `void copiarMatrizRenglon (`

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

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

Parameters

m	Matriz destino
r	Renglon origen
pos	Fila de inicio

Definition at line 162 of file mensajes.c.

4.66.3.6 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 154 of file mensajes.c.

4.66.3.7 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 240 of file mensajes.c.

4.66.3.8 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 281 of file mensajes.c.

4.66.3.9 printRenglon() `void printRenglon (`

```
renglon_t r )
```

imprime dos renglon en consola (para debuggear)

Parameters

r	renglon a imprimir

Definition at line 105 of file mensajes.c.

4.66.3.10 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 228 of file mensajes.c.

4.66.3.11 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 336 of file mensajes.c.

4.66.3.12 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 142 of file mensajes.c.

4.66.3.13 renglonIzquierdoLibre() `bool renglonIzquierdoLibre (`

```
mensaje_t * msj )
```

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

Parameters

r	Renglon a chequear

Returns

true

false

Definition at line 168 of file mensajes.c.

4.66.3.14 renglonNot() `void renglonNot (`

```
renglon_t r )
```

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

Parameters

r	Renglon a invertir

Definition at line 148 of file mensajes.c.

4.66.3.15 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 136 of file mensajes.c.

4.66.3.16 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 124 of file mensajes.c.

4.66.3.17 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 130 of file mensajes.c.

4.66.3.18 uintARenglon()

```
void uintARenglon (
    uint16_t n,
    renglon_t r )
```

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

Parameters

n	entero no signado de 16 bits
r	Renglon

Definition at line 205 of file mensajes.c.

4.67 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
00029 /*******************************************************************************
00030  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00031  ******************************************************************************/
00032
00033 /*******************************************************************************
00034  * VARIABLES WITH GLOBAL SCOPE
00035  ******************************************************************************/
00036
00037 // +ej: unsigned int anio_actual;+
00038
00039 /*******************************************************************************
00040  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00041  ******************************************************************************/
00042
00049 static int getLongitud(char a);
00050
00051 /*******************************************************************************
00052  * ROM CONST VARIABLES WITH FILE LEVEL SCOPE
00053  ******************************************************************************/
00054
00055 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, 5}; // sin contar Ñ (+ espacio + numeros + FULL)
00056
00057 /*******************************************************************************
00058  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00059  ******************************************************************************/
00060
00061 static uint16_t char_index[][TAM_RENGLON] = {{0x4000, 0xA000, 0xE000, 0xA000, 0xA000}, // A
00062                                              {0xC000, 0xA000, 0xC000, 0xA000, 0xC000},
00063                                              {0x4000, 0xA000, 0x8000, 0xA000, 0x4000},
00064                                              {0xC000, 0xA000, 0xA000, 0xA000, 0xC000},
00065                                              {0xE000, 0x8000, 0xC000, 0x8000, 0xE000},
00066                                              {0xE000, 0x8000, 0xC000, 0x8000, 0x8000},
00067                                              {0x6000, 0x8000, 0xA000, 0xA000, 0x6000},
00068                                              {0xA000, 0xA000, 0xE000, 0xA000, 0xA000},
00069                                              {0x8000, 0x8000, 0x8000, 0x8000, 0x8000},
00070                                              {0x2000, 0x2000, 0xA000, 0xA000, 0x4000},
00071                                              {0xA000, 0xA000, 0xC000, 0xA000, 0xA000},
00072                                              {0x8000, 0x8000, 0x8000, 0x8000, 0xE000},
00073                                              {0x8800, 0xD800, 0xA800, 0xA800, 0x8800},
00074                                              {0x9000, 0xD000, 0xB000, 0x9000, 0x9000},
00075                                              {0x4000, 0xA000, 0xA000, 0xA000, 0x4000},
00076                                              {0xC000, 0xA000, 0xA000, 0xC000, 0x8000},
00077                                              {0x4000, 0xA000, 0xA000, 0xA000, 0x6000},
00078                                              {0xC000, 0xA000, 0xC000, 0xA000, 0xA000},
00079                                              {0x6000, 0x8000, 0x4000, 0x2000, 0xC000},
00080                                              {0xE000, 0x4000, 0x4000, 0x4000, 0x4000},
00081                                              {0xA000, 0xA000, 0xA000, 0xA000, 0x4000},
00082                                              {0xA000, 0xA000, 0xA000, 0x4000, 0x4000},
00083                                              {0x8800, 0xA800, 0xA800, 0x5000, 0x5000},
00084                                              {0xA000, 0xA000, 0x4000, 0xA000, 0xA000},
00085                                              {0xA000, 0xA000, 0x4000, 0x4000, 0x4000},
00086                                              {0xE000, 0x2000, 0x4000, 0x8000, 0xE000}, // Z
00087                                              {0x0000, 0x0000, 0x0000, 0x0000, 0x0000}, // espacio
00088                                              {0x4000, 0xA000, 0xA000, 0xA000, 0x4000}, // 0
00089                                              {0x4000, 0xC000, 0x4000, 0x4000, 0xE000},
00090                                              {0x4000, 0xA000, 0x2000, 0x4000, 0xE000},
00091                                              {0xC000, 0x2000, 0x4000, 0x2000, 0xC000},
00092                                              {0xA000, 0xA000, 0xE000, 0x2000, 0x2000},
00093                                              {0xE000, 0x8000, 0xC000, 0x2000, 0xC000},
00094                                              {0x4000, 0x8000, 0xC000, 0xA000, 0xC000},
```

```
00095                                                 {0xE000, 0x2000, 0x2000, 0x4000, 0x4000},
00096                                                 {0x4000, 0xA000, 0x4000, 0xA000, 0x4000},
00097                                                 {0x4000, 0xA000, 0x6000, 0x2000, 0x4000},  // 9
00098                                                 {0xF800, 0xF800, 0xF800, 0xF800, 0xF800}}; // TODO (FULL)
00099
00100 /*******************************************************************************
00101  *******************************************************************************
00102                         GLOBAL FUNCTION DEFINITIONS
00103  *******************************************************************************
00104  *******************************************************************************/
00105 void printRenglon(renglon_t r)
00106 {
00107   for (int i = 0; i < TAM_RENGLON; i++, putchar('\n'))
00108   {
00109     for (int j = 0; j < 2 * CANT_FILAS; j++)
00110     {
00111       putchar((r[i].completo & (0x80000000 >> j)) ? 'x' : '.');
00112       if (j == CANT_FILAS - 1)
00113         putchar('|');
00114     }
00115   }
00116 }
00117
00118 void borrarRenglon(renglon_t r)
00119 {
00120   for (int i = 0; i < TAM_RENGLON; i++)
00121     r[i].completo = 0;
00122 }
00123
00124 void renglonShiftDer(renglon_t r, uint16_t s)
00125 {
00126   for (int i = 0; i < TAM_RENGLON; i++)
00127     r[i].completo >>= s;
00128 }
00129
00130 void renglonShiftIzq(renglon_t r, uint16_t s)
00131 {
00132   for (int i = 0; i < TAM_RENGLON; i++)
00133     r[i].completo <<= s;
00134 }
00135
00136 void renglonOr(renglon_t r, renglon_t s)
00137 {
00138   for (int i = 0; i < TAM_RENGLON; i++)
00139     r[i].completo |= s[i].completo;
00140 }
00141
00142 void renglonAnd(renglon_t r, renglon_t s)
00143 {
00144   for (int i = 0; i < TAM_RENGLON; i++)
00145     r[i].completo &= s[i].completo;
00146 }
00147
00148 void renglonNot(renglon_t r)
00149 {
00150   for (int i = 0; i < TAM_RENGLON; i++)
00151     r[i].completo = ~r[i].completo;
00152 }
00153
00154 void copiarRenglon(renglon_t r1, const renglon_t r2)
00155 {
00156   for (int i = 0; i < TAM_RENGLON; i++)
00157   {
00158     r1[i].completo = r2[i].completo;
00159   }
00160 }
00161
00162 void copiarMatrizRenglon(matriz_t m, const renglon_t r, int pos)
00163 {
00164   for (int i = 0; i < TAM_RENGLON; i++)
00165     m[pos + i] = r[i].mitad_izq;
00166 }
00167
00168 bool renglonIzquierdoLibre(mensaje_t *msj)
00169 {
00170   for (int i = 0; i < TAM_RENGLON; i++)
00171     if ((msj->renglon)[i].mitad_izq)
00172       return false;
00173   return true;
00174 }
00175
00176 void charARenglon(char c, renglon_t r)
00177 {
00178   int indice;
00179
00180   if (c == ' ' || !c)
00181     indice = INDEX_ESPACIO;
```

```
00182   else if ('0' <= c && c <= '9')
00183     indice = INDEX_CERO + c - '0';
00184   else if ('A' <= c && c <= 'Z')
00185     indice = c - 'A';
00186   else
00187     indice = INDEX_FULL;
00188 
00189   for (int i = 0; i < TAM_RENGLON; i++)
00190   {
00191     r[i].mitad_izq = char_index[indice][i];
00192     r[i].mitad_der = 0;
00193   }
00194 }
00195 
00196 void charAMatriz(char c, matriz_t m, const int coord[])
00197 {
00198   limpiarMatriz(m);
00199   renglon_t r;
00200   charARenglon(c, r);
00201   renglonShiftDer(r, coord[0]);
00202   copiarMatrizRenglon(m, r, coord[1]);
00203 }
00204 
00205 void uintARenglon(uint16_t n, renglon_t r)
00206 {
00207   renglon_t renglon_aux;
00208   int j = 0, resto = CANT_COLUMNAS;
00209   uint16_t div = 10000;
00210   while (n % div == n)
00211     div /= 10;
00212   while (n)
00213   {
00214     uint16_t aux = n % div;
00215     n = (n - aux) / div;
00216     resto = CANT_COLUMNAS - j - longitudes[INDEX_CERO + n];
00217     if (resto < 0)
00218       break;
00219     charARenglon(n + '0', renglon_aux);
00220     renglonShiftDer(renglon_aux, j);
00221     renglonOr(r, renglon_aux);
00222     j += longitudes[INDEX_CERO + n] + 1;
00223     n = aux;
00224     div /= 10;
00225   }
00226 }
00227 
00228 void reemplazarLetra(renglon_t r, char c, int j)
00229 {
00230   renglon_t full, letra;
00231   charARenglon(c, letra);
00232   renglonShiftDer(letra, j);
00233   charARenglon(PEDIR_FULL, full);
00234   renglonShiftDer(full, j);
00235   renglonNot(full);
00236   renglonAnd(r, full);
00237   renglonOr(r, letra);
00238 }
00239 
00240 mensaje_t mensaje(char *msj, int pos, bool repetir)
00241 {
00242   mensaje_t mensaje = {.posicion = pos, .habilitacion = true, .repetir_msj = repetir};
00243   borrarRenglon(mensaje.renglon);
00244   strcpy(mensaje.msj, msj);
00245   int longitud_parcial = strlen(mensaje.msj);
00246 
00247   int j = 0; // a partir de donde voy a escribir la proxima vez
00248   int i;
00249   for (i = 0; i < longitud_parcial; i++)
00250   {                  // rellena el mensaje por primera vez
00251     char c = msj[i]; // el caracter que debo escribir
00252 
00253     renglon_t letra;
00254     charARenglon(c, letra); // letra contiene la letra provisoria pasada a renglón
00255 
00256     int ancho = getLongitud(c);
00257     if (2 * CANT_COLUMNAS - j - ancho < 0)
00258       break; // lo que me quedaría libre si escribo
00259 
00260     renglonShiftDer(letra, j);         // muevo la letra sobre renglon
00261     renglonOr(mensaje.renglon, letra); // escribo en el renglon
00262     j += ancho + 1;                    // dejo un espacio entre letra y letra
00263   }
00264 
00265   mensaje.index = i;
00266   mensaje.j = j;
00267 
00268   mensaje.mover_texto = (i < longitud_parcial) || (j > CANT_COLUMNAS + 1); // veo si entra el mensaje
```

```
      en el renglon izquierdo
00269
00270   if (mensaje.mover_texto)
00271   {
00272     strcat(mensaje.msj, "     ");
00273     mensaje.longitud = strlen(mensaje.msj);
00274   }
00275   else
00276     mensaje.longitud = longitud_parcial;
00277
00278   return mensaje;
00279 }
00280
00281 void moverMensaje(mensaje_t *msj)
00282 {
00283   if (!(msj->mover_texto) || !(msj->habilitacion)) // tengo permitido mover el mensaje?
00284     return;
00285
00286   renglonShiftIzq(msj->renglon, 1);
00287   msj->j--;
00288
00289   if (!(msj->repetir_msj) && msj->index >= msj->longitud) // tengo que repetirlo cuando termine de
      recorrerlo?
00290   {
00291     if (renglonIzquierdoLibre(msj)) // termine de pasar el mensaje?
00292       msj->habilitacion = false;
00293     return;
00294   }
00295
00296   char c = (msj->msj)[msj->index];
00297   int ancho = getLongitud(c);
00298
00299   if (2 * CANT_COLUMNAS - msj->j - ancho < 0) // me queda espacio para poner la siguiente letra?
00300     return;
00301
00302   // pongo la siguiente letra
00303   renglon_t letra;
00304   charARenglon(c, letra);
00305   renglonShiftDer(letra, msj->j);
00306   renglonOr(msj->renglon, letra);
00307   msj->j += ancho + 1; // dejo un espacio entre letra y letra
00308   if (++msj->index >= msj->longitud && msj->repetir_msj)
00309     msj->index = 0;
00310 }
00311
00312 void concatenarLetraMensaje(char c, mensaje_t *msj)
00313 {
00314   if (msj->index == L_MAX - 1)
00315     return;
00316
00317   (msj->msj)[msj->index++] = c;
00318   (msj->msj)[msj->index] = '\0';
00319   msj->longitud++;
00320   msj->j += getLongitud(c) + 1;
00321
00322   int ancho = 5; // debo considerar el peor caso por si después se va a reemplazar la letra
00323
00324   while (CANT_COLUMNAS - msj->j - ancho < 0) // el margen estará antes
00325   {
00326     renglonShiftIzq(msj->renglon, 1);
00327     msj->j--;
00328   }
00329
00330   renglon_t letra;
00331   charARenglon('A', letra);
00332   renglonShiftDer(letra, msj->j);
00333   renglonOr(msj->renglon, letra);
00334 }
00335
00336 void reemplazarUltLetraMensaje(char c, mensaje_t *msj)
00337 {
00338   (msj->msj)[msj->index - 1] = c;
00339
00340   reemplazarLetra(msj->renglon, c, msj->j);
00341 }
00342
00343 int getLongitud(char c)
00344 {
00345   if (c == ' ' || !c)
00346     return longitudes[INDEX_ESPACIO];
00347   else if ('0' <= c && c <= '9')
00348     return longitudes[INDEX_CERO + c - '0'];
00349   else if ('A' <= c && c <= 'Z')
00350     return longitudes[c - 'A'];
00351   else
00352     return longitudes[INDEX_FULL];
00353 }
```

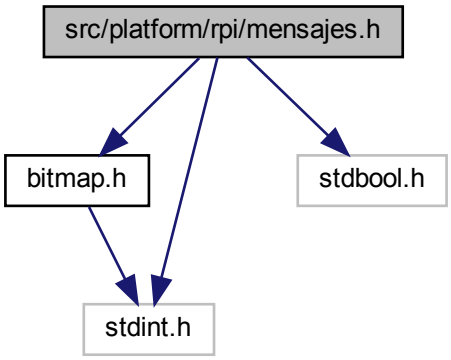
```
00354
00355 /*******************************************************************************
00356  *******************************************************************************
00357                         LOCAL FUNCTION DEFINITIONS
00358  *******************************************************************************
00359  ******************************************************************************/
```

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

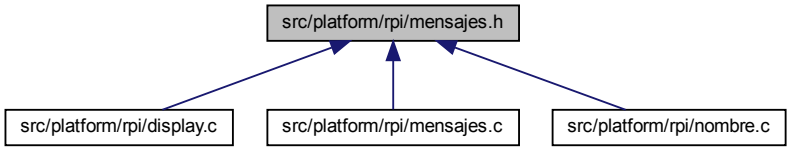
Encabezado de mensajes, con definiciones sobre tipos de datos y funciones.

```
#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

4.68.1 Detailed Description

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

Authors

AGRIPPINO, ALVAREZ, CASTRO, HEIR

Copyright

Copyright (c) 2022 ~ Ingeniería Electrónica ~ ITBA

Definition in file mensajes.h.

4.68.2 Macro Definition Documentation

4.68.2.1 L_MAX `#define L_MAX 64`

Definition at line 29 of file mensajes.h.

4.68.2.2 POS_MSJ1 `#define POS_MSJ1 2`

Definition at line 26 of file mensajes.h.

4.68.2.3 POS_MSJ2 `#define POS_MSJ2 9`

Definition at line 27 of file mensajes.h.

4.68.2.4 POS_MSJ3 `#define POS_MSJ3 5`

Definition at line 28 of file mensajes.h.

4.68.2.5 TAM_RENGLON `#define TAM_RENGLON 5`

Definition at line 24 of file mensajes.h.

4.68.3 Function Documentation

4.68.3.1 borrarRenglon() `void borrarRenglon ( renglon_t r )`

elimina el contenido del Renglon

Parameters

r	Renglon a limpiar

Definition at line 118 of file mensajes.c.

4.68.3.2 charAMatriz()

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

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

Parameters

c	caracter
m	Matriz
x	
y	

Definition at line 196 of file mensajes.c.

4.68.3.3 charARenglon()

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

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

Parameters

c	caracter
r	Renglon

Definition at line 176 of file mensajes.c.

4.68.3.4 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 312 of file mensajes.c.

4.68.3.5 copiarMatrizRenglon()

```
void copiarMatrizRenglon (
        matriz_t m,
        const renglon_t r,
        int pos )
```

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

Parameters

m	Matriz destino
r	Renglon origen
pos	Fila de inicio

Definition at line 162 of file mensajes.c.

4.68.3.6 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 154 of file mensajes.c.

4.68.3.7 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 240 of file mensajes.c.

4.68.3.8 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 281 of file mensajes.c.

4.68.3.9 printRenglon()

```
void printRenglon (
        renglon_t r )
```

imprime dos renglon en consola (para debuggear)

Parameters

r	renglon a imprimir

Definition at line 105 of file mensajes.c.

4.68.3.10 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 228 of file mensajes.c.

4.68.3.11 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 336 of file mensajes.c.

4.68.3.12 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 142 of file mensajes.c.

4.68.3.13 renglonIzquierdoLibre() `bool renglonIzquierdoLibre (`

```
mensaje_t * msj )
```

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

Parameters

r	Renglon a chequear

Returns

true

false

Definition at line 168 of file mensajes.c.

4.68.3.14 renglonNot() `void renglonNot (`

```
renglon_t r )
```

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

Parameters

r	Renglon a invertir

Definition at line 148 of file mensajes.c.

4.68.3.15 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 136 of file mensajes.c.

4.68.3.16 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 124 of file mensajes.c.

4.68.3.17 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 130 of file mensajes.c.

4.68.3.18 uintARenglon()

```
void uintARenglon (
        uint16_t n,
        renglon_t r )
```

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

Parameters

n	entero no signado de 16 bits
r	Renglon

Definition at line 205 of file mensajes.c.

4.69 mensajes.h

Go to the documentation of this file.

```
00001
00008 #ifndef _MENSAJES_H_
00009 #define _MENSAJES_H_
00010
00011 /*****************************************************************************
00012  * INCLUDE HEADER FILES
00013  *****************************************************************************/
00014
00015 #include "bitmap.h"
00016
00017 #include <stdint.h>
00018 #include <stdbool.h>
00019
00020 /*****************************************************************************
00021  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00022  *****************************************************************************/
00023
00024 #define TAM_RENGLON 5
00025
00026 #define POS_MSJ1 2
00027 #define POS_MSJ2 9
00028 #define POS_MSJ3 5
00029 #define L_MAX 64
00030
00031 /*****************************************************************************
00032  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00033  *****************************************************************************/
00034 typedef union
00035 {
00036   uint32_t completo;
00037   struct
00038   {
00039     uint16_t mitad_der;
00040     uint16_t mitad_izq;
00041   };
00042 } renglon_t[5];
00043
00044 typedef struct Mensaje
00045 {
00046   char msj[L_MAX];
00047   int posicion;
00048   int index;
00049   int longitud;
00050   int j;
00051   bool habilitacion;
00052   bool mover_texto;
00053   bool repetir_msj;
00054   renglon_t renglon;
00055 } mensaje_t;
00056
```

```
00057 /*****************************************************************************
00058  * VARIABLE PROTOTYPES WITH GLOBAL SCOPE
00059  *****************************************************************************/
00060
00061 // +ej: extern unsigned int anio_actual;+
00062
00063 /*****************************************************************************
00064  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00065  *****************************************************************************/
00066
00072 void printRenglon(renglon_t r);
00073
00079 void borrarRenglon(renglon_t r);
00080
00087 void renglonShiftDer(renglon_t r, uint16_t s);
00088
00095 void renglonShiftIzq(renglon_t r, uint16_t s);
00096
00103 void renglonOr(renglon_t r, renglon_t s);
00104
00111 void renglonAnd(renglon_t r, renglon_t s);
00112
00118 void renglonNot(renglon_t r);
00119
00126 void copiarRenglon(renglon_t r1, const renglon_t r2);
00127
00135 void copiarMatrizRenglon(matriz_t m, const renglon_t r, int pos);
00136
00144 bool renglonIzquierdoLibre(mensaje_t *msj);
00145
00152 void charARenglon(char c, renglon_t r);
00153
00162 void charAMatriz(char c, matriz_t m, const int coord[]);
00163
00170 void uintARenglon(uint16_t n, renglon_t r); // copia un número a renglon hasta que se acabe el número
      o el renglon (lo 1 q ocurra)
00171
00179 void reemplazarLetra(renglon_t r, char c, int j);
00180
00189 mensaje_t mensaje(char *msj, int pos, bool repetir);
00190
00196 void moverMensaje(mensaje_t *msj);
00197
00204 void concatenarLetraMensaje(char c, mensaje_t *msj);
00205
00212 void reemplazarUltLetraMensaje(char c, mensaje_t *msj);
00213
00214 /*****************************************************************************
00215  *****************************************************************************/
00216
00217 #endif // _MENSAJES_H_
```

4.70 audio.c

```
00001 /*
00002  * Simple-SDL2-Audio
00003  *
00004  * Copyright 2016 Jake Besworth
00005  *
00006  * Licensed under the Apache License, Version 2.0 (the "License");
00007  * you may not use this file except in compliance with the License.
00008  * You may obtain a copy of the License at
00009  *
00010  *     http://www.apache.org/licenses/LICENSE-2.0
00011  *
00012  * Unless required by applicable law or agreed to in writing, software
00013  * distributed under the License is distributed on an "AS IS" BASIS,
00014  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00015  * See the License for the specific language governing permissions and
00016  * limitations under the License.
00017  *
00018  */
00019 #include <stdint.h>
00020 #include <stdio.h>
00021 #include <stdlib.h>
00022
00023 #include <SDL2/SDL.h>
00024
00025 #include "audio.h"
00026
00027 /*
00028  * Native WAVE format
00029  *
00030  * On some GNU/Linux you can identify a files properties using:
```

```
00031  *      mplayer -identify music.wav
00032  *
00033  * On some GNU/Linux to convert any music to this or another specified format use:
00034  *      ffmpeg -i in.mp3 -acodec pcm_s16le -ac 2 -ar 48000 out.wav
00035  */
00036 /* SDL_AudioFormat of files, such as s16 little endian */
00037 #define AUDIO_FORMAT AUDIO_S16LSB
00038
00039 /* Frequency of the file */
00040 #define AUDIO_FREQUENCY 48000
00041
00042 /* 1 mono, 2 stereo, 4 quad, 6 (5.1) */
00043 #define AUDIO_CHANNELS 2
00044
00045 /* Specifies a unit of audio data to be used at a time. Must be a power of 2 */
00046 #define AUDIO_SAMPLES 4096
00047
00048 /* Max number of sounds that can be in the audio queue at anytime, stops too much mixing */
00049 #define AUDIO_MAX_SOUNDS 25
00050
00051 /* The rate at which the volume fades when musics transition. The higher number indicates music fading
      faster */
00052 #define AUDIO_MUSIC_FADE_VALUE 2
00053
00054 /* Flags OR'd together, which specify how SDL should behave when a device cannot offer a specific
      feature
00055  * If flag is set, SDL will change the format in the actual audio file structure (as opposed to
      gDevice->want)
00056  *
00057  * Note: If you're having issues with Emscripten / EMCC play around with these flags
00058  *
00059  * 0                                 Allow no changes
00060  * SDL_AUDIO_ALLOW_FREQUENCY_CHANGE  Allow frequency changes (e.g. AUDIO_FREQUENCY is 48k, but
      allow files to play at 44.1k
00061  * SDL_AUDIO_ALLOW_FORMAT_CHANGE     Allow Format change (e.g. AUDIO_FORMAT may be S32LSB, but
      allow wave files of S16LSB to play)
00062  * SDL_AUDIO_ALLOW_CHANNELS_CHANGE   Allow any number of channels (e.g. AUDIO_CHANNELS being 2,
      allow actual 1)
00063  * SDL_AUDIO_ALLOW_ANY_CHANGE        Allow all changes above
00064  */
00065 #define SDL_AUDIO_ALLOW_CHANGES SDL_AUDIO_ALLOW_ANY_CHANGE
00066
00067 /*
00068  * Definition for the game global sound device
00069  *
00070  */
00071 typedef struct privateAudioDevice
00072 {
00073     SDL_AudioDeviceID device;
00074     SDL_AudioSpec want;
00075     uint8_t audioEnabled;
00076 } PrivateAudioDevice;
00077
00078 /* File scope variables to persist data */
00079 static PrivateAudioDevice * gDevice;
00080 static uint32_t gSoundCount;
00081
00082 /*
00083  * Add a music to the queue, addAudio wrapper for music due to fade
00084  *
00085  * @param newAudio       New Audio to add
00086  *
00087  */
00088 static void addMusic(Audio * root, Audio * newAudio);
00089
00090 /*
00091  * Wrapper function for playMusic, playSound, playMusicFromMemory, playSoundFromMemory
00092  *
00093  * @param filename      Provide a filename to load WAV from, or NULL if using FromMemory
00094  * @param audio         Provide an Audio object if copying from memory, or NULL if using a filename
00095  * @param sound         1 if looping (music), 0 otherwise (sound)
00096  * @param volume        See playSound for explanation
00097  *
00098  */
00099 static inline void playAudio(const char * filename, Audio * audio, uint8_t loop, int volume);
00100
00101 /*
00102  * Add a sound to the end of the queue
00103  *
00104  * @param root           Root of queue
00105  * @param newAudio       New Audio to add
00106  *
00107  */
00108 static void addAudio(Audio * root, Audio * newAudio);
00109
00110 /*
00111  * Audio callback function for OpenAudioDevice
```

```
00112  *
00113  * @param userdata      Points to linked list of sounds to play, first being a placeholder
00114  * @param stream        Stream to mix sound into
00115  * @param len           Length of sound to play
00116  *
00117  */
00118 static inline void audioCallback(void * userdata, uint8_t * stream, int len);
00119
00120 void playSound(const char * filename, int volume)
00121 {
00122     playAudio(filename, NULL, 0, volume);
00123 }
00124
00125 void playMusic(const char * filename, int volume)
00126 {
00127     playAudio(filename, NULL, 1, volume);
00128 }
00129
00130 void playSoundFromMemory(Audio * audio, int volume)
00131 {
00132     playAudio(NULL, audio, 0, volume);
00133 }
00134
00135 void playMusicFromMemory(Audio * audio, int volume)
00136 {
00137     playAudio(NULL, audio, 1, volume);
00138 }
00139
00140 void initAudio(void)
00141 {
00142     Audio * global;
00143     gDevice = (PrivateAudioDevice *) calloc(1, sizeof(PrivateAudioDevice));
00144     gSoundCount = 0;
00145
00146     if(gDevice == NULL)
00147     {
00148         fprintf(stderr, "[%s: %d]Fatal Error: Memory c-allocation error\n", __FILE__, __LINE__);
00149         return;
00150     }
00151
00152     gDevice->audioEnabled = 0;
00153
00154     if(!(SDL_WasInit(SDL_INIT_AUDIO) & SDL_INIT_AUDIO))
00155     {
00156         fprintf(stderr, "[%s: %d]Error: SDL_INIT_AUDIO not initialized\n", __FILE__, __LINE__);
00157         return;
00158     }
00159
00160     SDL_memset(&(gDevice->want), 0, sizeof(gDevice->want));
00161
00162     (gDevice->want).freq = AUDIO_FREQUENCY;
00163     (gDevice->want).format = AUDIO_FORMAT;
00164     (gDevice->want).channels = AUDIO_CHANNELS;
00165     (gDevice->want).samples = AUDIO_SAMPLES;
00166     (gDevice->want).callback = audioCallback;
00167     (gDevice->want).userdata = calloc(1, sizeof(Audio));
00168
00169     global = (Audio *) (gDevice->want).userdata;
00170
00171     if(global == NULL)
00172     {
00173         fprintf(stderr, "[%s: %d]Error: Memory allocation error\n", __FILE__, __LINE__);
00174         return;
00175     }
00176
00177     global->buffer = NULL;
00178     global->next = NULL;
00179
00180     /* want.userdata = newAudio; */
00181     if((gDevice->device = SDL_OpenAudioDevice(NULL, 0, &(gDevice->want), NULL,
      SDL_AUDIO_ALLOW_CHANGES)) == 0)
00182     {
00183         fprintf(stderr, "[%s: %d]Warning: failed to open audio device: %s\n", __FILE__, __LINE__,
      SDL_GetError());
00184     }
00185     else
00186     {
00187         /* Set audio device enabled global flag */
00188         gDevice->audioEnabled = 1;
00189
00190         /* Unpause active audio stream */
00191         unpauseAudio();
00192     }
00193 }
00194
00195 void endAudio(void)
00196 {
```

```
00197     if(gDevice->audioEnabled)
00198     {
00199         pauseAudio();
00200 
00201         freeAudio((Audio *) (gDevice->want).userdata);
00202 
00203         /* Close down audio */
00204         SDL_CloseAudioDevice(gDevice->device);
00205     }
00206 
00207     free(gDevice);
00208 }
00209 
00210 void pauseAudio(void)
00211 {
00212     if(gDevice->audioEnabled)
00213     {
00214         SDL_PauseAudioDevice(gDevice->device, 1);
00215     }
00216 }
00217 
00218 void unpauseAudio(void)
00219 {
00220     if(gDevice->audioEnabled)
00221     {
00222         SDL_PauseAudioDevice(gDevice->device, 0);
00223     }
00224 }
00225 
00226 void freeAudio(Audio * audio)
00227 {
00228     Audio * temp;
00229 
00230     while(audio != NULL)
00231     {
00232         if(audio->free == 1)
00233         {
00234             SDL_FreeWAV(audio->bufferTrue);
00235         }
00236 
00237         temp = audio;
00238         audio = audio->next;
00239 
00240         free(temp);
00241     }
00242 }
00243 
00244 Audio * createAudio(const char * filename, uint8_t loop, int volume)
00245 {
00246     Audio * newAudio = (Audio *) calloc(1, sizeof(Audio));
00247 
00248     if(newAudio == NULL)
00249     {
00250         fprintf(stderr, "[%s: %d]Error: Memory allocation error\n", __FILE__, __LINE__);
00251         return NULL;
00252     }
00253 
00254     if(filename == NULL)
00255     {
00256         fprintf(stderr, "[%s: %d]Warning: filename: NULL\n", __FILE__, __LINE__);
00257         return NULL;
00258     }
00259 
00260     newAudio->next = NULL;
00261     newAudio->loop = loop;
00262     newAudio->fade = 0;
00263     newAudio->free = 1;
00264     newAudio->volume = volume;
00265 
00266     if(SDL_LoadWAV(filename, &(newAudio->audio), &(newAudio->bufferTrue), &(newAudio->lengthTrue)) ==
      NULL)
00267     {
00268         fprintf(stderr, "[%s: %d]Warning: failed to open wave file: %s error: %s\n", __FILE__,
     __LINE__, filename, SDL_GetError());
00269         free(newAudio);
00270         return NULL;
00271     }
00272 
00273     newAudio->buffer = newAudio->bufferTrue;
00274     newAudio->length = newAudio->lengthTrue;
00275     (newAudio->audio).callback = NULL;
00276     (newAudio->audio).userdata = NULL;
00277 
00278     return newAudio;
00279 }
00280 
00281 static inline void playAudio(const char * filename, Audio * audio, uint8_t loop, int volume)
```

```
00282 {
00283     Audio * newAudio;
00284
00285     /* Check if audio is enabled */
00286     if(!gDevice->audioEnabled)
00287     {
00288         return;
00289     }
00290
00291     /* If sound, check if under max number of sounds allowed, else don't play */
00292     if(loop == 0)
00293     {
00294         if(gSoundCount >= AUDIO_MAX_SOUNDS)
00295         {
00296             return;
00297         }
00298         else
00299         {
00300             gSoundCount++;
00301         }
00302     }
00303
00304     /* Load from filename or from Memory */
00305     if(filename != NULL)
00306     {
00307         /* Create new music sound with loop */
00308         newAudio = createAudio(filename, loop, volume);
00309     }
00310     else if(audio != NULL)
00311     {
00312         newAudio = (Audio *) malloc(sizeof(Audio));
00313
00314         if(newAudio == NULL)
00315         {
00316             fprintf(stderr, "[%s: %d]Fatal Error: Memory allocation error\n", __FILE__, __LINE__);
00317             return;
00318         }
00319
00320         memcpy(newAudio, audio, sizeof(Audio));
00321
00322         newAudio->volume = volume;
00323         newAudio->loop = loop;
00324         newAudio->free = 0;
00325     }
00326     else
00327     {
00328         fprintf(stderr, "[%s: %d]Warning: filename and Audio parameters NULL\n", __FILE__, __LINE__);
00329         return;
00330     }
00331
00332     /* Lock callback function */
00333     SDL_LockAudioDevice(gDevice->device);
00334
00335     if(loop == 1)
00336     {
00337         addMusic((Audio *) (gDevice->want).userdata, newAudio);
00338     }
00339     else
00340     {
00341         addAudio((Audio *) (gDevice->want).userdata, newAudio);
00342     }
00343
00344     SDL_UnlockAudioDevice(gDevice->device);
00345
00346 }
00347
00348 static void addMusic(Audio * root, Audio * newAudio)
00349 {
00350     uint8_t musicFound = 0;
00351     Audio * rootNext = root->next;
00352
00353     /* Find any existing musics, 0, 1 or 2 and fade them out */
00354     while(rootNext != NULL)
00355     {
00356         /* Phase out any current music */
00357         if(rootNext->loop == 1 && rootNext->fade == 0)
00358         {
00359             if(musicFound)
00360             {
00361                 rootNext->length = 0;
00362                 rootNext->volume = 0;
00363             }
00364
00365             rootNext->fade = 1;
00366         }
00367         /* Set flag to remove any queued up music in favour of new music */
00368         else if(rootNext->loop == 1 && rootNext->fade == 1)
```

```
00369         {
00370             musicFound = 1;
00371         }
00372 
00373         rootNext = rootNext->next;
00374     }
00375 
00376     addAudio(root, newAudio);
00377 }
00378 
00379 static inline void audioCallback(void * userdata, uint8_t * stream, int len)
00380 {
00381     Audio * audio = (Audio *) userdata;
00382     Audio * previous = audio;
00383     int tempLength;
00384     uint8_t music = 0;
00385 
00386     /* Silence the main buffer */
00387     SDL_memset(stream, 0, len);
00388 
00389     /* First one is place holder */
00390     audio = audio->next;
00391 
00392     while(audio != NULL)
00393     {
00394         if(audio->length > 0)
00395         {
00396             if(audio->fade == 1 && audio->loop == 1)
00397             {
00398                 music = 1;
00399 
00400                 if(audio->volume > 0)
00401                 {
00402                     if(audio->volume - AUDIO_MUSIC_FADE_VALUE < 0)
00403                     {
00404                         audio->volume = 0;
00405                     }
00406                     else
00407                     {
00408                         audio->volume -= AUDIO_MUSIC_FADE_VALUE;
00409                     }
00410                 }
00411                 else
00412                 {
00413                     audio->length = 0;
00414                 }
00415             }
00416 
00417             if(music && audio->loop == 1 && audio->fade == 0)
00418             {
00419                 tempLength = 0;
00420             }
00421             else
00422             {
00423                 tempLength = ((uint32_t) len > audio->length) ? audio->length : (uint32_t) len;
00424             }
00425 
00426             SDL_MixAudioFormat(stream, audio->buffer, AUDIO_FORMAT, tempLength, audio->volume);
00427 
00428             audio->buffer += tempLength;
00429             audio->length -= tempLength;
00430 
00431             previous = audio;
00432             audio = audio->next;
00433         }
00434         else if(audio->loop == 1 && audio->fade == 0)
00435         {
00436             audio->buffer = audio->bufferTrue;
00437             audio->length = audio->lengthTrue;
00438         }
00439         else
00440         {
00441             previous->next = audio->next;
00442 
00443             if(audio->loop == 0)
00444             {
00445                 gSoundCount--;
00446             }
00447 
00448             audio->next = NULL;
00449             freeAudio(audio);
00450 
00451             audio = previous->next;
00452         }
00453     }
00454 }
00455 
```

```
00456 static void addAudio(Audio * root, Audio * newAudio)
00457 {
00458     if(root == NULL)
00459     {
00460         return;
00461     }
00462
00463     while(root->next != NULL)
00464     {
00465         root = root->next;
00466     }
00467
00468     root->next = newAudio;
00469 }
```

4.71 audio.h

```
00001 /*
00002  * Simple-SDL2-Audio
00003  *
00004  * Copyright 2016 Jake Besworth
00005  *
00006  * Licensed under the Apache License, Version 2.0 (the "License");
00007  * you may not use this file except in compliance with the License.
00008  * You may obtain a copy of the License at
00009  *
00010  *     http://www.apache.org/licenses/LICENSE-2.0
00011  *
00012  * Unless required by applicable law or agreed to in writing, software
00013  * distributed under the License is distributed on an "AS IS" BASIS,
00014  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00015  * See the License for the specific language governing permissions and
00016  * limitations under the License.
00017  *
00018  */
00019
00020 /*
00021  * audio.h
00022  *
00023  * All audio related functions go here
00024  *
00025  */
00026 #ifndef SIMPLE_AUDIO_
00027 #define SIMPLE_AUDIO_
00028
00029 #ifdef __cplusplus
00030 extern "C"
00031 {
00032 #endif
00033
00034 #include <SDL2/SDL.h>
00035
00036 /*
00037  * Queue structure for all loaded sounds
00038  *
00039  */
00040 typedef struct sound
00041 {
00042     uint32_t length;
00043     uint32_t lengthTrue;
00044     uint8_t * bufferTrue;
00045     uint8_t * buffer;
00046     uint8_t loop;
00047     uint8_t fade;
00048     uint8_t free;
00049     uint8_t volume;
00050
00051     SDL_AudioSpec audio;
00052
00053     struct sound * next;
00054 } Audio;
00055
00056 /*
00057  * Create a Audio object
00058  *
00059  * @param filename      Filename for the WAVE file to load
00060  * @param loop          0 ends after playing once (sound), 1 repeats and fades when other music added
       (music)
00061  * @param volume        Volume, read playSound()
00062  *
00063  * @return returns a new Audio or NULL on failure, you must call freeAudio() on return Audio
00064  *
00065  */
00066 Audio * createAudio(const char * filename, uint8_t loop, int volume);
```

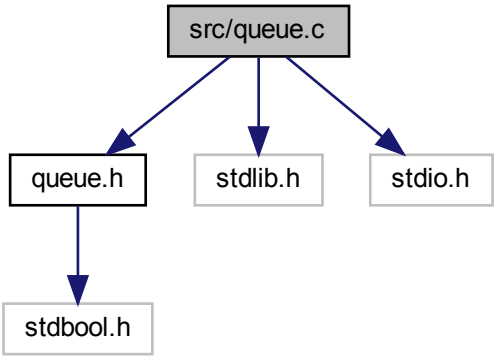
```
00067
00068 /*
00069  * Frees as many chained Audios as given
00070  *
00071  * @param audio     Chain of sounds to free
00072  *
00073  */
00074 void freeAudio(Audio * audio);
00075
00076 /*
00077  * Play a wave file currently must be S16LE format 2 channel stereo
00078  *
00079  * @param filename      Filename to open, use getAbsolutePath
00080  * @param volume        Volume 0 - 128. SDL_MIX_MAXVOLUME constant for max volume
00081  *
00082  */
00083 void playSound(const char * filename, int volume);
00084
00085 /*
00086  * Plays a new music, only 1 at a time plays
00087  *
00088  * @param filename      Filename of the WAVE file to load
00089  * @param volume        Volume read playSound for moree
00090  *
00091  */
00092 void playMusic(const char * filename, int volume);
00093
00094 /*
00095  * Plays a sound from a createAudio object (clones), only 1 at a time plays
00096  * Advantage to this method is no more disk reads, only once, data is stored and constantly reused
00097  *
00098  * @param audio         Audio object to clone and use
00099  * @param volume        Volume read playSound for moree
00100  *
00101  */
00102 void playSoundFromMemory(Audio * audio, int volume);
00103
00104 /*
00105  * Plays a music from a createAudio object (clones), only 1 at a time plays
00106  * Advantage to this method is no more disk reads, only once, data is stored and constantly reused
00107  *
00108  * @param audio         Audio object to clone and use
00109  * @param volume        Volume read playSound for moree
00110  *
00111  */
00112 void playMusicFromMemory(Audio * audio, int volume);
00113
00114 /*
00115  * Free all audio related variables
00116  * Note, this needs to be run even if initAudio fails, because it frees the global audio device
00117  *
00118  */
00119 void endAudio(void);
00120
00121 /*
00122  * Initialize Audio Variable
00123  *
00124  */
00125 void initAudio(void);
00126
00127 /*
00128  * Pause audio from playing
00129  *
00130  */
00131 void pauseAudio(void);
00132
00133 /*
00134  * Unpause audio from playing
00135  *
00136  */
00137 void unpauseAudio(void);
00138
00139 #ifdef __cplusplus
00140 }
00141 #endif
00142
00143 #endif
```

4.72 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 <stdio.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.

4.72.1 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.

4.72.2 Function Documentation

4.72.2.1 destruirQueue() `void destruirQueue (`

`void )`

Destruye la cola de eventos.

Definition at line 100 of file queue.c.

4.72.2.2 queueInsertar() `void queueInsertar (`

`event_t nuevo )`

Agrega un evento a la cola.

Definition at line 57 of file queue.c.

4.72.2.3 queueSiguienteEvento() `event_t queueSiguienteEvento (`

`void )`

Devuelve el siguiente evento de la cola.

Returns

event_t

Definition at line 86 of file queue.c.

4.72.2.4 queueVacia() `bool queueVacia (`

`void )`

Chequea si la cola está vacía.

Returns

true Vacía

false No vacía

Definition at line 81 of file queue.c.

4.73 queue.c

Go to the documentation of this file.

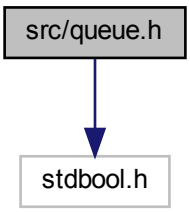
```
00001
00012 /*******************************************************************************
00013  * INCLUDE HEADER FILES
00014  ******************************************************************************/
00015
00016 #include "queue.h"
00017
00018 #include <stdlib.h>
00019 #include <stdio.h>
00020
00021
00022 /*******************************************************************************
00023  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00024  ******************************************************************************/
00025
00026 //Estructura de la queue
00027 typedef struct nodeT
00028 {
00029     event_t data;
00030     struct nodeT *next;
00031 } node_t;
00032
00033
00034 /*******************************************************************************
00035  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00036  ******************************************************************************/
00037
00042 static void borrarElemento(void);
00043
00044
00045 /*******************************************************************************
00046  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00047  ******************************************************************************/
00048
00049 static node_t *front = NULL, *back = NULL;
00050
00051 /*******************************************************************************
00052  *******************************************************************************
00053                         GLOBAL FUNCTION DEFINITIONS
00054  *******************************************************************************
00055  ******************************************************************************/
00056
00057 void queueInsertar(event_t nuevo)
00058 {
00059     node_t *temp = (node_t *)malloc(sizeof(node_t));
00060
00061     temp->data = nuevo;
00062
00063     if (front == NULL)
00064     {
00065         front = temp;
00066         front->next = NULL;
00067     }
00068     else if (back == NULL)
00069     {
00070         back = temp;
00071         front->next = back;
00072         back->next = NULL;
00073     }
00074     else
00075     {
00076         back->next = temp;
00077         back = back->next;
00078     }
00079 }
00080
00081 bool queueVacia(void)
00082 {
00083     return front == NULL;
00084 }
00085
00086 event_t queueSiguienteEvento(void)
00087 {
00088     if (front == NULL)
00089     {
00090         return NADA;
00091     }
00092     else
00093     {
00094         event_t r = front->data;
00095         borrarElemento();
00096         return r;
00097     }
```

```
00098 }
00099
00100 void destruirQueue(void)
00101 {
00102     while (front != NULL)
00103     {
00104         queueSiguienteEvento();
00105     }
00106 }
00107
00108 /*****************************************************************************
00109  *****************************************************************************
00110                         LOCAL FUNCTION DEFINITIONS
00111  *****************************************************************************
00112  *****************************************************************************/
00113
00114 static void borrarElemento(void)
00115 {
00116     if (front != NULL)
00117     {
00118         node_t *temp = front;
00119         front = front->next;
00120         free(temp);
00121         if (front == NULL)
00122         {
00123             back = NULL;
00124         }
00125     }
00126 }
00127
00128 /*
00129 static void printQueue(){
00130   node_t *temp = front;
00131   while(temp != back){
00132     printf("%d, ", temp->data);
00133     temp = temp->next;
00134   }
00135 }
00136 */
```

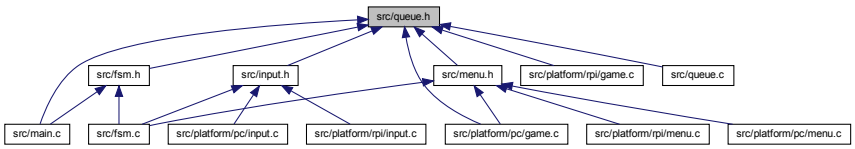
4.74 src/queue.h File Reference

Header del modulo queue. Prototipos de funciones de interaccion con la cola de eventos.

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



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



Typedefs

- typedef int event_t

Enumerations

- enum **eventos** { **NADA** = -1 , **SALIR** = 0 , **GAME_OVER** , **FORCE_SALIR** }
- enum **eventos_tecla** {
  NO_MOVER = -1 , **ESC** = 59 , **BORRAR** = 63 , **ENTER** = 67 ,
  IZDA = 82 , **DCHA** , **ARRIBA** , **ABAJO** }

Functions

- void queueInsertar (event_t)

  Agrega un evento a la cola.
- bool queueVacia (void)

  Chequea si la cola está vacía.
- event_t queueSiguienteEvento (void)

  Devuelve el siguiente evento de la cola.
- void destruirQueue (void)

  Destruye la cola de eventos.

4.74.1 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.

4.74.2 Typedef Documentation

4.74.2.1 event_t `typedef int event_t`

Definition at line 28 of file queue.h.

4.74.3 Enumeration Type Documentation

4.74.3.1 eventos `enum eventos`

Definition at line 31 of file queue.h.

4.74.3.2 eventos_tecla `enum eventos_tecla`

Definition at line 40 of file queue.h.

4.74.4 Function Documentation

4.74.4.1 destruirQueue() `void destruirQueue ( void )`

Destruye la cola de eventos.

Definition at line 100 of file queue.c.

4.74.4.2 queueInsertar() `void queueInsertar ( event_t *nuevo* )`

Agrega un evento a la cola.

Definition at line 57 of file queue.c.

4.74.4.3 queueSiguienteEvento() `event_t queueSiguienteEvento ( void )`

Devuelve el siguiente evento de la cola.

Returns

event_t

Definition at line 86 of file queue.c.

4.74.4.4 queueVacia() `bool queueVacia ( void )`

Chequea si la cola está vacía.

Returns

true Vacía

false No vacía

Definition at line 81 of file queue.c.

4.75 queue.h

Go to the documentation of this file.

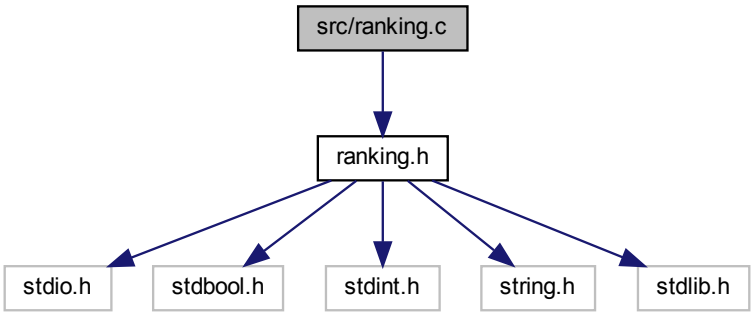
```
00001
00012 // https://stackoverflow.com/questions/3536153/c-dynamically-growing-array
00013
00014 #ifndef _QUEUE_H_
00015 #define _QUEUE_H_
00016
00017 /*****************************************************************************
00018  * INCLUDE HEADER FILES
00019  *****************************************************************************/
00020
00021 #include <stdbool.h>
00022
00023 /*****************************************************************************
00024  * ENUMERATIONS AND STRUCTURES AND TYPEDEFS
00025  *****************************************************************************/
00026
00027 // Tipo de dato para eventos
00028 typedef int event_t;
00029
00030 // Eventos posibles principales
00031 enum eventos
00032 {
00033   NADA = -1,
00034   SALIR = 0,
00035   GAME_OVER,
00036   FORCE_SALIR
00037 };
00038
00039 // Eventos posibles de interacción en el juego
00040 enum eventos_tecla
00041 {
00042   NO_MOVER = -1,
00043   ESC = 59,
00044   BORRAR = 63,
00045   ENTER = 67,
00046   IZDA = 82,
00047   DCHA,
00048   ARRIBA,
```

```
00049   ABAJO
00050 };
00051
00052 /*****************************************************************************
00053  * FUNCTION PROTOTYPES WITH GLOBAL SCOPE
00054  *****************************************************************************/
00055
00060 void queueInsertar(event_t);
00061
00068 bool queueVacia(void);
00069
00075 event_t queueSiguienteEvento(void);
00076
00081 void destruirQueue(void);
00082
00083 /*****************************************************************************
00084  *****************************************************************************/
00085
00086 #endif // _QUEUE_H_
```

4.76 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 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

4.76.1 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.

4.76.2 Macro Definition Documentation

4.76.2.1 MAX_LEN `#define MAX_LEN 100`

Definition at line 24 of file ranking.c.

4.76.3 Function Documentation

4.76.3.1 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 101 of file ranking.c.

4.76.3.2 desiniciarRanking() `void desiniciarRanking ( void )`

Desinicializa el sistema de ranking, actualizando el archivo correspondiente.

Definition at line 142 of file ranking.c.

4.76.3.3 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.

4.76.3.4 getRankingLineas() `uint getRankingLineas ( void )`

Devuelve cantidad de renglones del ranking.

Returns

int Renglones

Definition at line 197 of file ranking.c.

4.76.3.5 getRankingNombres() `char ** getRankingNombres ( void )`

Devuelve array de nombres de jugadores.

Returns

char**

Definition at line 202 of file ranking.c.

4.76.3.6 getRankingPuntos() `unsigned long long * getRankingPuntos ( void )`

Devuelve array de puntos de jugadores.

Returns

unsigned long long*

Definition at line 207 of file ranking.c.

4.76.3.7 iniciarRanking() `void iniciarRanking ( void )`

Inicializa el sistema de ranking.

Definition at line 85 of file ranking.c.

4.76.3.8 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 155 of file ranking.c.

4.76.4 Variable Documentation

4.76.4.1 handlerRanking `FILE* handlerRanking = NULL`

Definition at line 64 of file ranking.c.

4.76.4.2 handlerTemp `FILE* handlerTemp = NULL`

Definition at line 66 of file ranking.c.

4.77 ranking.c

Go to the documentation of this file.

```
00001
00013 /*****************************************************************************
00014  * INCLUDE HEADER FILES
00015  *****************************************************************************/
00016
00017 #include "ranking.h"
00018
00019 /*****************************************************************************
00020  * CONSTANT AND MACRO DEFINITIONS USING #DEFINE
00021  *****************************************************************************/
00022
00023 // Largo maximo de una linea del txt
00024 #define MAX_LEN 100
00025
00026 /*****************************************************************************
00027  * FUNCTION PROTOTYPES FOR PRIVATE FUNCTIONS WITH FILE LEVEL SCOPE
00028  *****************************************************************************/
00029
00034 static void recargarRanking(void);
00035
00040 static void ordenarRanking(void);
00041
00046 static void writeRanking(void);
00047
00052 static void createRankingFile(void);
00053
00054 /*****************************************************************************
00055  * STATIC VARIABLES AND CONST VARIABLES WITH FILE LEVEL SCOPE
00056  *****************************************************************************/
00057
00058 // Nombre del archivo de ranking
00059 static char *strRanking = "ranking.txt";
00060 // Nombre del archivo temporal
00061 static char *strTemp = "temp.txt";
00062
00063 // Handler del archivo de ranking
00064 FILE *handlerRanking = NULL;
00065 // Handler del archivo temporal
00066 FILE *handlerTemp = NULL;
00067
00068 // Punteros a nombres
00069 static char **names = NULL;
00070 // Puntero a scores
00071 static unsigned long long *scores = NULL;
00072
00073 // String temporal
00074 static char tempStr[MAX_LEN];
00075
00076 // Contador de lineas del txt
00077 static uint lineNumber = 0;
00078
00079 /*****************************************************************************
00080  *****************************************************************************
00081                         GLOBAL FUNCTION DEFINITIONS
00082  *****************************************************************************
```

```
00083  ***************************************************************************/
00084
00085 void iniciarRanking(void)
00086 {
00087   lineNumber = 0;
00088
00089   createRankingFile();
00090
00091   if ((handlerRanking = fopen(strRanking, "r")) == NULL)
00092   {
00093     printf("Error opening ranking.txt");
00094   }
00095
00096   recargarRanking();
00097
00098   fclose(handlerRanking);
00099 }
00100
00101 void actualizarRanking(char *name, unsigned long long score)
00102 {
00103   int i;
00104   bool player_exists;
00105
00106   // Veo si el jugador esta en el ranking
00107   for (i = 0, player_exists = false; i < lineNumber && !player_exists; i++)
00108   {
00109     if (!lineNumber)
00110       break;
00111
00112     // Si el nombre coincide...
00113     if (strcmp(names[i], name) == 0)
00114     {
00115       // Actualiza el score
00116       scores[i] = score;
00117       player_exists = true;
00118     }
00119   }
00120
00121   // Si el jugador no existe en el ranking, lo agrego al final
00122   if (!player_exists)
00123   {
00124     // Reservo memoria para un puntero
00125     names = (char **)realloc(names, sizeof(char *) * (lineNumber + 1));
00126     // Reservo memoria para el nombre
00127     names[lineNumber] = (char *)calloc(strlen(name), sizeof(char));
00128     // Asigno nombre
00129     strcpy(names[lineNumber], name);
00130
00131     // Reservo memoria para un score
00132     scores = (unsigned long long *)realloc(scores, sizeof(unsigned long long) * (lineNumber + 1));
00133     // Asigno score
00134     scores[lineNumber] = score;
00135
00136     lineNumber++;
00137   }
00138
00139   ordenarRanking();
00140 }
00141
00142 void desiniciarRanking(void)
00143 {
00144   // Escribe al archivo
00145   writeRanking();
00146
00147   // Liberacion de memoria
00148   int i;
00149   for (i = 0; i < lineNumber; i++)
00150     free(names[i]);
00151
00152   free(names);
00153 }
00154
00155 bool verificarJugadorRanking(char *name)
00156 {
00157   int i;
00158   bool exists;
00159
00160   // Ranking vacio
00161   if (!lineNumber)
00162     return false;
00163
00164   for (i = 0, exists = false; i < lineNumber && !exists; i++)
00165   {
00166     // Si el nombre coincide...
00167     if (strcmp(names[i], name) == 0)
00168       exists = true;
00169   }
```

```
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;
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   if (!exists)
00192     score = 0;
00193
00194   return score;
00195 }
00196
00197 uint getRankingLineas(void)
00198 {
00199   return lineNumber;
00200 }
00201
00202 char **getRankingNombres(void)
00203 {
00204   return names;
00205 }
00206
00207 unsigned long long *getRankingPuntos(void)
00208 {
00209   return scores;
00210 }
00211
00212 /*****************************************************************************
00213  *****************************************************************************
00214                         LOCAL FUNCTION DEFINITIONS
00215  *****************************************************************************
00216  *****************************************************************************/
00217
00218 static void recargarRanking(void)
00219 {
00220   lineNumber = 0;
00221
00222   while (fgets(tempStr, MAX_LEN, handlerRanking) != NULL)
00223   {
00224     // Remove the trailing newline character
00225     if (strchr(tempStr, '\n'))
00226       tempStr[strlen(tempStr) - 1] = '\0';
00227
00228     // Puntero al nombre
00229     char *tempPtr = strtok(tempStr, " ");
00230     // Reservo memoria para un puntero
00231     names = (char **)realloc(names, sizeof(char *) * (lineNumber + 1));
00232     // Reservo memoria para el nombre
00233     names[lineNumber] = (char *)calloc(strlen(tempPtr), sizeof(char));
00234     // Copia nombre
00235     strcpy(names[lineNumber], tempPtr);
00236
00237     // Puntero al score
00238     tempPtr = strtok(NULL, " ");
00239     // Reservo memoria para un score
00240     scores = (unsigned long long *)realloc(scores, sizeof(unsigned long long) * (lineNumber + 1));
00241     // Copia score
00242     scores[lineNumber] = strtoul(tempPtr, NULL, 10);
00243
00244     lineNumber++;
00245   }
00246 }
00247
00248 static void ordenarRanking(void)
00249 {
00250   int i, j;
00251   unsigned long long tempScore;
00252
00253   for (i = 0; i < (lineNumber - 1); i++)
00254   {
00255     for (j = 0; j < (lineNumber - i - 1); j++)
00256     {
```

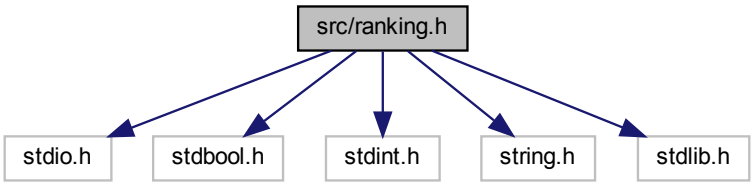
```
00257       // Si el primer score es menor, o si es igual al siguiente pero predomina orden alfabetico...
00258       if ((scores[j] < scores[j + 1]) || ((scores[j] == scores[j + 1]) && (strcmp(names[j], names[j +
      1]) > 0)))
00259       {
00260         // Backup del menor
00261         strcpy(tempStr,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

4.78 src/ranking.h File Reference

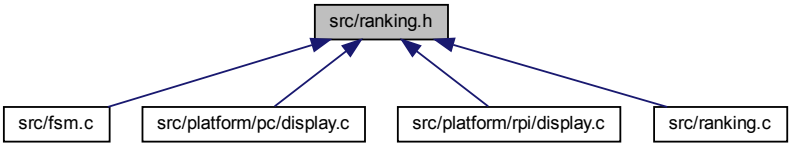
Header del modulo ranking. Prototipos de funciones de interaccion con el ranking de jugadores. Permite trabajar con el txt correspondiente fácilmente,.

```
#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.

4.78.1 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.

4.78.2 Macro Definition Documentation

4.78.2.1 DEFAULT_PLAYER_NAME `#define DEFAULT_PLAYER_NAME "PLAYER"`

Definition at line 30 of file ranking.h.

4.78.3 Function Documentation

4.78.3.1 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 101 of file ranking.c.

4.78.3.2 desiniciarRanking() `void desiniciarRanking (`

```
void )
```

Desinicializa el sistema de ranking, actualizando el archivo correspondiente.

Definition at line 142 of file ranking.c.

4.78.3.3 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.

4.78.3.4 getRankingLineas() `uint getRankingLineas ( void )`

Devuelve cantidad de renglones del ranking.

Returns

int Renglones

Definition at line 197 of file ranking.c.

4.78.3.5 getRankingNombres() `char ** getRankingNombres ( void )`

Devuelve array de nombres de jugadores.

Returns

char∗∗

Definition at line 202 of file ranking.c.

4.78.3.6 getRankingPuntos() `unsigned long long * getRankingPuntos ( void )`

Devuelve array de puntos de jugadores.

Returns

unsigned long long∗

Definition at line 207 of file ranking.c.

4.78.3.7 iniciarRanking() void iniciarRanking (void)

Inicializa el sistema de ranking.

Definition at line 85 of file ranking.c.

4.78.3.8 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 155 of file ranking.c.

4.79 ranking.h

Go to the documentation of this file.

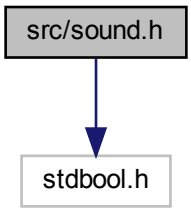
```
00001
00013 #ifndef _RANKING_H_
00014 #define _RANKING_H_
00015
00016 /*****************************************************************************
00017  * INCLUDE HEADER FILES
00018  *****************************************************************************/
00019
00020 #include <stdio.h>
00021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

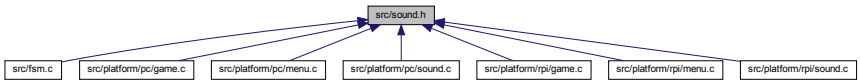
4.80 src/sound.h File Reference

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.

4.80.1 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.

4.80.2 Enumeration Type Documentation

4.80.2.1 efectos `enum efectos`

Definition at line 39 of file sound.h.

4.80.2.2 musica `enum musica`

Definition at line 27 of file sound.h.

4.80.3 Function Documentation

4.80.3.1 destruirSonido() `void destruirSonido (`

`void )`

Desinicializa el sonido de la plataforma.

Definition at line 31 of file sound.c.

4.80.3.2 iniciarSonido() `bool iniciarSonido (`

`void )`

Inicializa el sonido de la plataforma.

Returns

true Exito

false Error

Definition at line 26 of file sound.c.

4.80.3.3 pausarMusica() `void pausarMusica (`

`void )`

Pausa la musica actual.

Definition at line 35 of file sound.c.

4.80.3.4 reproducirEfecto() `void reproducirEfecto (`

`int efecto )`

Pone a reproducir un efecto dado.

Parameters

int	num efectos

Definition at line 75 of file sound.c.

4.80.3.5 reproducirMusica() `void reproducirMusica (`

`int musica )`

Pone a reproducir una musica dada.

Parameters

int	enum musica

Definition at line 40 of file sound.c.

4.81 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