Super Mario Pro2

Generated by Doxygen 1.14.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 pro2 Namespace Reference

Classes

- struct DoubPt
- struct Pt
- struct Rect
- class TextWriter
- class Window

Typedefs

- typedef std::vector< std::vector< std::vector< std::string > > > Font

  Vector de caràcters. Els caràcters són matrius de strings. Cada element de la matriu representa un pixel a pintar, i cada string diferent està mapejada a la paleta amb el color que li correspon.
- typedef std::map< std::string, int > Palette

  Mapa de strings a colors.
- typedef std::map< char, int > Charset

  Mapa de caràcters a l'índex de l'sprite de la font que li correspon.
- typedef std::vector< std::vector< int > > Sprite
- typedef uint32_t Color

Enumerations

- enum ModKey { Ctrl = 1 , Shift = 2 , Alt = 4 , Meta = 8 }
- enum Keys {
  Space = 32 , Backspace = 8 , Delete = 127 , End = 5 ,
  Escape = 27 , Home = 2 , Insert = 26 , PageDown = 4 ,
  PageUp = 3 , Return = 10 , Tab = 9 , Up = 17 ,
  Down = 18 , Right = 19 , Left = 20 }

Functions

- std::pair< bool, int > resolve_collision_vertical (const Rect &prev, Rect curr, const Rect &block)

  Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat des de sobre i -1 altrament.
- std::pair< bool, int > resolve_collision_horizontal (const Rect &prev, Rect curr, const Rect &block)

  Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat d'esquerra a dreta i -1 altrament.
- bool operator< (const Pt &a, const Pt &b)

  Compara dos punts del pla.
- Pt round_dpt (const DoubPt &a)

  Retorna un pro2::Pt amb els valors truncats d'un pro2::DoubPt.
- bool check_collision (const Rect &a, const Rect &b)

  Retorna true si s'intersecccionen els dos pro2::Rect.
- std::ifstream read_file (std::string fname)

  Carrega un fitxer de text com a 'stream'.
- Palette read_colors (std::ifstream &stream)

  Genera una paleta de colors a partir d'un fitxer de text.
- Font read_sprites (std::ifstream &stream)

  Genera una font (vector de sprites de tots els caràcters) a partir d'un fitxer de text Els fitxers de tipus font tenen la següent estructura: A la primera línia hi ha 3 'int' amb 'count' (nombre de sprites), 'height' i 'width' (dels sprites) A continuació hi ha tots els sprites amb els caràcters.
- Charset read_charset (std::ifstream &stream)

  Carrega el charset (els caràcters disponibles i l'ordre corresponent) de la font a partir d'un fitxer de text.
- Sprite color_sprite (const std::vector< std::vector< std::string > > &sprite, const Palette &colors)

  Pinta el caràcter de la font amb una paleta de colors.
- void paint_char (Window &window, Pt &pos, Sprite sprite, int &size)

  Pinta a la pantalla un caràcter amb una posició i tamany.
- std::vector< std::string > split_lines (std::string text)

  *Separa un string multilínia (separades per '
  ') en un vector de strings per cada línia.*
- double random_double (int min, int max, int precision)

  Retorna un double entre (min, max), amb una precisó de n digits.
- void paint_hline (Window &window, int xini, int xfin, int y, Color color=white)

  Dibuja una línea horizontal en la ventana.
- void paint_vline (Window &window, int x, int yini, int yfin, Color color=white)

  Dibuja una línea vertical en la ventana.
- void paint_rect (Window &window, Rect rect, Color color, int brush_sz)

  Dibuixa un rectangle 'Rect'.
- void paint_rect_fill (Window &window, Rect rect, Color color)

  Dibuixa i emplena un rectangle 'Rect'.
- void paint_rect_fill_transparent (Window &window, Rect rect, Color color, double transp)

  Dibuixa i emplena un rectangle 'Rect' amb transparència.
- void paint_pixel_transparent (Window &window, Pt pos, Color color, double transp)

  Pinta un pixel amb color i transparència.
- void paint_sprite (Window &window, Pt orig, const Sprite &sprite, bool mirror)

  Dibuixa una imatge/textura a la finestra a partir d'una posició

Variables

- const Color black = 0x00000000
- const Color red = 0x00ff0000
- const Color green = 0x0000ff00
- const Color blue = 0x000000ff
- const Color yellow = 0x00ffff00
- const Color magenta = 0x00ff00ff
- const Color cyan = 0x0000ffff
- const Color white = 0x00ffffff

4.1.1 Typedef Documentation

4.1.1.1 Charset

```
typedef std::map<char, int> pro2::Charset
```

Mapa de caràcters a l'índex de l'sprite de la font que li correspon.

4.1.1.2 Color

```
typedef uint32_t pro2::Color
```

4.1.1.3 Font

```
typedef std::vector<std::vector<std::vector<std::string> > > pro2::Font
```

Vector de caràcters. Els caràcters són matrius de strings. Cada element de la matriu representa un pixel a pintar, i cada string diferent està mapejada a la paleta amb el color que li correspon.

4.1.1.4 Palette

```
typedef std::map<std::string, int> pro2::Palette
```

Mapa de strings a colors.

4.1.1.5 Sprite

```
typedef std::vector<std::vector<int> > pro2::Sprite
```

4.1.2 Enumeration Type Documentation

4.1.2.1 Keys

```
enum pro2::Keys
```

Enumerado con los códigos de las teclas que se pueden pasar al método `Window::is_key_down` para consultar el estado de una tecla.

Enumerator

Space	
Backspace	
Delete	
End	
Escape	
Home	
Insert	
PageDown	
PageUp	
Return	
Tab	
Up	
Down	
Right	
Left	

4.1.2.2 ModKey

```
enum pro2::ModKey
```

Enumerado para las 4 teclas de control: Ctrl, Shift, Alt, y Meta.

Enumerator

Ctrl	
Shift	
Alt	
Meta	

4.1.3 Function Documentation

4.1.3.1 check_collision()

```
bool pro2::check_collision (
            const Rect & a,
            const Rect & b)  [inline]
```

Retorna true si s'intersecccionen els dos pro2::Rect.

4.1.3.2 color_sprite()

```
Sprite pro2::color_sprite (
            const std::vector< std::vector< std::string > > & sprite,
            const Palette & colors)
```

Pinta el caràcter de la font amb una paleta de colors.

Parameters

sprite	Sprite del caràcter amb els templates del color
colors	Paleta de colors

4.1.3.3 operator<()

```
bool pro2::operator< (
            const Pt & a,
            const Pt & b)  [inline]
```

Compara dos punts del pla.

La comparació és necessària per poder fer servir `Pt` com la clau d'un `map`. La comparació utilitza primer la coordenada `x` (com si fos més "important"), i, quan les `x`s són iguals, la coordenada `y`.

4.1.3.4 paint_char()

```
void pro2::paint_char (
            Window & window,
            Pt & pos,
            Sprite sprite,
            int & size)
```

Pinta a la pantalla un caràcter amb una posició i tamany.

Parameters

window	Finestra a on dibuixar
pos	Posició a on dibuixar
sprite	Sprite del caràcter pintat a dibuixar
size	Gruix amb el que es dibuixarà el caràcter

4.1.3.5 paint_hline()

```
void pro2::paint_hline (
            Window & window,
            int xini,
            int xfin,
            int y,
            Color color = white)
```

Dibuja una línea horizontal en la ventana.

Parameters

window	Ventana en la que se dibuja la línea.
xini	Coordenada x inicial.
xfin	Coordenada x final.
y	Coordenada y.
color	Color de la línea (opcional, si no se pone se asume `white`).

4.1.3.6 paint_pixel_transparent()

```
void pro2::paint_pixel_transparent (
            Window & window,
            Pt pos,
            Color color,
            double transp)
```

Pinta un pixel amb color i transparència.

Parameters

window	Finestra a la que pintar
pos	Posició del pixel
color	Color
transp	Transparència (del 0 al 1) Amb transp=0 el rectangle és totalment opac, amb transp=1 totalment transparent (invisible)

4.1.3.7 paint_rect()

```
void pro2::paint_rect (
            Window & window,
            Rect rect,
            Color color,
            int brush_sz)
```

Dibuixa un rectangle 'Rect'.

Parameters

window	Finestra a la que pintar
rect	Rectangle a pintar
color	Color de les línies
brush_sz	Gruix de les línies

4.1.3.8 paint_rect_fill()

```
void pro2::paint_rect_fill (
            Window & window,
            Rect rect,
            Color color)
```

Dibuixa i emplena un rectangle 'Rect'.

Parameters

window	Finestra a la que pintar
rect	Rectangle a dibuixar
color	Color del rectangle

4.1.3.9 paint_rect_fill_transparent()

```
void pro2::paint_rect_fill_transparent (
            Window & window,
            Rect rect,
            Color color,
            double transp)
```

Dibuixa i emplena un rectangle 'Rect' amb transparència.

Parameters

window	Finestra a la que pintar
rect	Rectangle a dibuixar
color	Color del rectangle
transp	Transparència (del 0 al 1)

4.1.3.10 paint_sprite()

```
void pro2::paint_sprite (
            Window & window,
            Pt orig,
            const Sprite & sprite,
            bool mirror)
```

Dibuixa una imatge/textura a la finestra a partir d'una posició

Parameters

window	Finestra a la que pintar
orig	Origen (cantonada de dalt a l'esquerra) del rectangle que forma el *sprite*
sprite	Matriu de colors que representa la imatge (*sprite*).
mirror	Si cal pintar girar la textura horitzontalment

4.1.3.11 paint_vline()

```
void pro2::paint_vline (
            Window & window,
            int x,
            int yini,
            int yfin,
            Color color = white)
```

Dibuja una línea vertical en la ventana.

Parameters

window	Ventana en la que se dibuja la línea.
x	Coordenada x.
yini	Coordenada y inicial.
yfin	Coordenada y final.
color	Color de la línea (opcional, si no se pone se asume `white`).

4.1.3.12 random_double()

```
double pro2::random_double (
            int min,
            int max,
            int precision)
```

Retorna un double entre (min, max), amb una precisó de n digits.

Parameters

min	Valor mínim
max	Valor màxim
precision	Int de la forma 10^n

4.1.3.13 read_charset()

```
Charset pro2::read_charset (
            std::ifstream & stream)
```

Carrega el charset (els caràcters disponibles i l'ordre corresponent) de la font a partir d'un fitxer de text.

Parameters

stream	Stream del fitxer

4.1.3.14 read_colors()

```
Palette pro2::read_colors (
            std::ifstream & stream)
```

Genera una paleta de colors a partir d'un fitxer de text.

Parameters

stream	Stream del fitxer

4.1.3.15 read_file()

```
std::ifstream pro2::read_file (
            std::string fname)
```

Carrega un fitxer de text com a 'stream'.

Parameters

fname	Nom del fitxer

4.1.3.16 read_sprites()

```
Font pro2::read_sprites (
            std::ifstream & stream)
```

Genera una font (vector de sprites de tots els caràcters) a partir d'un fitxer de text Els fitxers de tipus font tenen la següent estructura: A la primera línia hi ha 3 'int' amb 'count' (nombre de sprites), 'height' i 'width' (dels sprites) A continuació hi ha tots els sprites amb els caràcters.

Parameters

stream	Stream del fitxer

4.1.3.17 resolve_collision_horizontal()

```
std::pair< bool, int > pro2::resolve_collision_horizontal (
            const Rect & prev,
            Rect curr,
            const Rect & block)
```

Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat d'esquerra a dreta i -1 altrament.

4.1.3.18 resolve_collision_vertical()

```
std::pair< bool, int > pro2::resolve_collision_vertical (
            const Rect & prev,
            Rect curr,
            const Rect & block)
```

Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat des de sobre i -1 altrament.

4.1.3.19 round_dpt()

```
Pt pro2::round_dpt (
            const DoubPt & a)  [inline]
```

Retorna un pro2::Pt amb els valors truncats d'un pro2::DoubPt.

4.1.3.20 split_lines()

```
std::vector< std::string > pro2::split_lines (
            std::string text)
```

Separa un string multilínia (separades per '
') en un vector de strings per cada línia.

Parameters

text	Text a dividir

4.1.4 Variable Documentation

4.1.4.1 black

```
const Color pro2::black = 0x00000000
```

4.1.4.2 blue

```
const Color pro2::blue = 0x000000ff
```

4.1.4.3 cyan

```
const Color pro2::cyan = 0x0000ffff
```

4.1.4.4 green

```
const Color pro2::green = 0x0000ff00
```

4.1.4.5 magenta

```
const Color pro2::magenta = 0x00ff00ff
```

4.1.4.6 red

```
const Color pro2::red = 0x00ff0000
```

4.1.4.7 white

```
const Color pro2::white = 0x00ffffff
```

4.1.4.8 yellow

```
const Color pro2::yellow = 0x00ffff00
```

Chapter 5

Class Documentation

5.1 Block Class Reference

`#include <block.hh>`

Public Member Functions

- Block (pro2::Pt pos={0, 0}, int type=0, int has_object=0)
- Block (const Block &other)
- void paint (pro2::Window &window, int anim_frame) const
- pro2::Pt pos () const
- int block_type () const

  Retorna el tipus de bloc que és.
- pro2::Rect get_rect () const
- std::vector< std::vector< int > > get_sprite (int anim_frame) const

  Retorna la matriu del sprite del frame de l'animació corresponent.
- int check_bumped (int state)

  Retorna:

Static Public Attributes

- static const std::vector< std::vector< std::vector< int > > > sprites

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Block() [1/2]

```
Block::Block (
            pro2::Pt pos = {0,0},
            int type = 0,
            int has_object = 0)  [inline]
```

5.1.1.2 Block() [2/2]

```
Block::Block (
            const Block & other)  [inline]
```

5.1.2 Member Function Documentation

5.1.2.1 block_type()

```
int Block::block_type () const  [inline]
```

Retorna el tipus de bloc que és.

Returns

- 0 si és un bloc de 'totxo'
- 1 si és un bloc 'interrogant'
- 2 si és un bloc 'activat'

5.1.2.2 check_bumped()

```
int Block::check_bumped (
            int state)  [inline]
```

Retorna:

- 0 si no passa res
- 1 si s'ha de trencar el bloc
- 2 si s'ha de crear una moneda (sense trencar)
- 3 si s'ha de crear un bolet (sense trencar)

5.1.2.3 get_rect()

```
pro2::Rect Block::get_rect () const  [inline]
```

5.1.2.4 get_sprite()

```
std::vector< std::vector< int > > Block::get_sprite (
            int anim_frame) const  [inline]
```

Retorna la matriu del sprite del frame de l'animació corresponent.

5.1.2.5 paint()

```
void Block::paint (
            pro2::Window & window,
            int anim_frame) const
```

5.1.2.6 pos()

```
pro2::Pt Block::pos () const  [inline]
```

5.1.3 Member Data Documentation

5.1.3.1 sprites

```
const std::vector<std::vector<std::vector<int> > > Block::sprites  [static]
```

The documentation for this class was generated from the following files:

- block.hh
- block.cc

5.2 Button Class Reference

Classe que implementa botons i la seva funcionalitat.

```
#include <start_screen.hh>
```

Public Member Functions

- Button (pro2::Rect rect, std::string text, int bg_normal=0xCD612E, int bg_selected=0xC97A55)
- pro2::Rect get_rect () const
- bool selected (pro2::Pt pos) const

  Retorna si el punter del ratolí és a sobre del botó
- void paint (pro2::Window &window, pro2::TextWriter &writer) const

5.2.1 Detailed Description

Classe que implementa botons i la seva funcionalitat.

```
Els botons poden ser clicats i quan el ratolí passa per sobre d'ells, poden canviar de color de fons (bg_norm
```

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Button()

```
Button::Button (
            pro2::Rect rect,
            std::string text,
            int bg_normal = 0xCD612E,
            int bg_selected = 0xC97A55)  [inline]
```

5.2.3 Member Function Documentation

5.2.3.1 get_rect()

```
pro2::Rect Button::get_rect () const  [inline]
```

5.2.3.2 paint()

```
void Button::paint (
            pro2::Window & window,
            pro2::TextWriter & writer) const
```

5.2.3.3 selected()

```
bool Button::selected (
            pro2::Pt pos) const  [inline]
```

Retorna si el punter del ratolí és a sobre del botó

The documentation for this class was generated from the following files:

- start_screen.hh
- start_screen.cc

5.3 Coin Class Reference

```
#include <coin.hh>
```

Public Member Functions

- Coin (pro2::Pt pos, pro2::DoubPt speed={0, 0}, pro2::DoubPt accel={0, -1}, pro2::DoubPt drag={0.075, 0.↩ 075})

  Les monedes s'inicialitzen amb una posició inicial i, opcionalment, amb velocitat, acceleració i coeficient de drag. En cas de deixar la velocitat, accel i drag en els seus valors per defecte, la moneda s'inicialitzarà estàtica, sense ser afectada per la gravetat.
- void paint (pro2::Window &window, int anim_frame) const

  Pinta el sprite de la moneda corresponent amb l'animació
- pro2::Pt pos () const
- pro2::Rect get_rect () const
- std::vector< std::vector< int > > get_sprite (int anim_frame) const

  Retorna la matriu del sprite del frame de l'animació corresponent.
- void set_y (int y)
- bool is_grounded () const
- void set_grounded (bool grounded)
- void toggle_grounded ()
- void update (pro2::Window &window, const std::set< Platform * > &platforms)

Static Public Attributes

- static const std::vector< std::vector< std::vector< int > > > sprites

5.3.1 Constructor & Destructor Documentation

5.3.1.1 Coin()

```
Coin::Coin (
            pro2::Pt pos,
            pro2::DoubPt speed = {0, 0},
            pro2::DoubPt accel = {0, -1},
            pro2::DoubPt drag = {0.075, 0.075})  [inline]
```

Les monedes s'inicialitzen amb una posició inicial i, opcionalment, amb velocitat, acceleració i coeficient de drag. En cas de deixar la velocitat, accel i drag en els seus valors per defecte, la moneda s'inicialitzarà estàtica, sense ser afectada per la gravetat.

Parameters

pos	Posició
speed	Velocitat
accel	Acceleració
drag	Coeficient de drag

5.3.2 Member Function Documentation

5.3.2.1 get_rect()

```
pro2::Rect Coin::get_rect () const  [inline]
```

5.3.2.2 get_sprite()

```
std::vector< std::vector< int > > Coin::get_sprite (
            int anim_frame) const  [inline]
```

Retorna la matriu del sprite del frame de l'animació corresponent.

5.3.2.3 is_grounded()

```
bool Coin::is_grounded () const  [inline]
```

5.3.2.4 paint()

```
void Coin::paint (
            pro2::Window & window,
            int anim_frame) const
```

Pinta el sprite de la moneda corresponent amb l'animació

Parameters

window	Finestra on es dibuixarà

5.3.2.5 pos()

```
pro2::Pt Coin::pos () const  [inline]
```

5.3.2.6 set_grounded()

```
void Coin::set_grounded (
            bool grounded)  [inline]
```

5.3.2.7 set_y()

```
void Coin::set_y (
            int y)  [inline]
```

5.3.2.8 toggle_grounded()

```
void Coin::toggle_grounded ()  [inline]
```

5.3.2.9 update()

```
void Coin::update (
            pro2::Window & window,
            const std::set< Platform * > & platforms)
```

5.3.3 Member Data Documentation

5.3.3.1 sprites

```
const std::vector<std::vector<std::vector<int> > > Coin::sprites  [static]
```

The documentation for this class was generated from the following files:

- coin.hh
- coin.cc

5.4 pro2::DoubPt Struct Reference

```
#include <geometry.hh>
```

Public Attributes

- double x = 0
- double y = 0

5.4.1 Member Data Documentation

5.4.1.1 x

```
double pro2::DoubPt::x = 0
```

5.4.1.2 y

```
double pro2::DoubPt::y = 0
```

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

- geometry.hh

5.5 fenster Struct Reference

```
#include <fenster.h>
```

Public Attributes

- const char * title
- const int width
- const int height
- uint32_t * buf
- int keys [256]
- int mod
- int x
- int y
- int mouse
- Display * dpy
- Window w
- GC gc
- XImage * img

5.5.1 Member Data Documentation

5.5.1.1 buf

```
uint32_t* fenster::buf
```

5.5.1.2 dpy

```
Display* fenster::dpy
```

5.5.1.3 gc

```
GC fenster::gc
```

5.5.1.4 height

```
const int fenster::height
```

5.5.1.5 img

```
XImage* fenster::img
```

5.5.1.6 keys

```
int fenster::keys[256]
```

5.5.1.7 mod

```
int fenster::mod
```

5.5.1.8 mouse

```
int fenster::mouse
```

5.5.1.9 title

```
const char* fenster::title
```

5.5.1.10 w

```
Window fenster::w
```

5.5.1.11 width

```
const int fenster::width
```

5.5.1.12 x

```
int fenster::x
```

5.5.1.13 y

```
int fenster::y
```

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

- fenster.h

5.6 Finder< T > Class Template Reference

```
#include <finder.hh>
```

Public Member Functions

- Finder (pro2::Rect range={0, 0, MAX_SZ, MAX_SZ}, pro2::Pt divider={MAX_SZ/NUM_DIVS, MAX_SZ/NUM_DIVS})
  Crea un objecte Finder< T > i inicialitza el contenidor amb les cel·les buides.
- void add (T *t)
  Afegeix un element al finder.
- void update (T *t)
  Actualitza les coordenades de l'objecte.
- void remove (T *t)
  Elimina un objecte del contenidor.
- void remove_and_delete (T *t)
  Elimina un objecte del contenidor i el borra.
- std::set< T * > query (pro2::Rect rect) const
  Retorna un set amb els punters que intersseccionen amb 'rect'.
- void AddFromList (std::list< T > &set)
  Afegeix tots els elements d'un set al finder.

5.6.1 Detailed Description

template<typename T>
class Finder< T >

La meva implementació de l'objecte finder consisteix en un contenidor, dividit en cel·les. Cada element s'afegirà a les cel·les que es corresponen amb les coordenades on es troba l'objecte.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Finder()

```
template<typename T>
Finder< T >::Finder (
            pro2::Rect range = {0,0,MAX_SZ,MAX_SZ},
            pro2::Pt divider = {MAX_SZ/NUM_DIVS, MAX_SZ/NUM_DIVS})  [inline]
```

Crea un objecte Finder<T> i inicialitza el contenidor amb les cel·les buides.

Precondition

Els objectes de tipus T han de tenir un mètode 'get_rect()' que retorni un pro2::Rect

Parameters

range	pro2::Rect amb {xmin, ymin, xmax, ymax} del rang de coordenades que admet
divider	pro2::Pt amb el tamany de cada divisió del contenidor

Els valors per defecte són {0,0,MAX_SZ,MAX_SZ} i {MAX_SZ/NUM_DIVS,MAX_SZ/NUM_DIVS}, que per defecte estan inicialitzades a MAX_SZ = 20000 i NUM_DIVS = 32 (valors emprats per a la solució del Jutge)

5.6.3 Member Function Documentation

5.6.3.1 add()

```
template<typename T>
void Finder< T >::add (
            T * t)  [inline]
```

Afegeix un element al finder.

Parameters

t	(T *t) Punter a l'element a afegir.

5.6.3.2 AddFromList()

```
template<typename T>
void Finder< T >::AddFromList (
            std::list< T > & set)  [inline]
```

Afegeix tots els elements d'un set al finder.

5.6.3.3 query()

```
template<typename T>
std::set< T * > Finder< T >::query (
            pro2::Rect rect) const  [inline]
```

Retorna un set amb els punters que interseccionen amb 'rect'.

Parameters

rect	pro2::Rect amb les coordenades a comprovar

5.6.3.4 remove()

```
template<typename T>
void Finder< T >::remove (
            T * t)  [inline]
```

Elimina un objecte del contenidor.

Parameters

T	*t Punter a l'objecte

Precondition

L'objecte apuntat *t ja és present al contenidor (s'ha afegit previament i no s'ha eliminat)

5.6.3.5 remove_and_delete()

```
template<typename T>
void Finder< T >::remove_and_delete (
            T * t)  [inline]
```

Elimina un objecte del contenidor i el borra.

Parameters

T	*t Punter a l'objecte

Precondition

L'objecte apuntat *t ja és present al contenidor (s'ha afegit previament i no s'ha eliminat)

5.6.3.6 update()

```
template<typename T>
void Finder< T >::update (
            T * t)  [inline]
```

Actualitza les coordenades de l'objecte.

Parameters

T	*t Punter a l'objecte

Precondition

L'objecte apuntat *t ja és present al contenidor (s'ha afegit previament i no s'ha eliminat)

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

- finder.hh

5.7 Game Class Reference

`#include <game.hh>`

Public Member Functions

- Game (int width, int height, pro2::TextWriter TW, pro2::Rect death_barrier=pro2::Rect{-1000, -2000, 100000, 400})
- void update (pro2::Window &window)
- void paint (pro2::Window &window)
- bool is_finished () const
- bool is_paused () const
- int exit_code () const

  Retorna 1 si el jugador ha mort, 2 si s'ha guanyat o 0 altrament.
- void spawn_coin (pro2::Pt pos, pro2::DoubPt vel)

  Crea una nova moneda i l'afegeix al finder.
- void anim_step ()

  Actualitza el comptador de l'animació per passar al següent sprite.

5.7.1 Constructor & Destructor Documentation

5.7.1.1 Game()

```
Game::Game (
            int width,
            int height,
            pro2::TextWriter TW,
            pro2::Rect death_barrier = pro2::Rect{-1000, -2000, 100000, 400})
```

5.7.2 Member Function Documentation

5.7.2.1 anim_step()

```
void Game::anim_step ()  [inline]
```

Actualitza el comptador de l'animació per passar al següent sprite.

5.7.2.2 exit_code()

```
int Game::exit_code () const  [inline]
```

Retorna 1 si el jugador ha mort, 2 si s'ha guanyat o 0 altrament.

5.7.2.3 is_finished()

```
bool Game::is_finished () const  [inline]
```

5.7.2.4 is_paused()

```
bool Game::is_paused () const  [inline]
```

5.7.2.5 paint()

```
void Game::paint (
            pro2::Window & window)
```

5.7.2.6 spawn_coin()

```
void Game::spawn_coin (
            pro2::Pt pos,
            pro2::DoubPt vel)
```

Crea una nova moneda i l'afegeix al finder.

5.7.2.7 update()

```
void Game::update (
            pro2::Window & window)
```

The documentation for this class was generated from the following files:

- game.hh
- game.cc

5.8 Interactable Class Reference

La classe Interactable engloba tot tipus d'objectes que tenen un comportament diferent amb els que el mario pot interactuar.

```
#include <interactables.hh>
```

Public Member Functions

- Interactable (pro2::Pt pos, int type)
- bool update (pro2::Window &window, const Finder< Platform > &platforms, const Finder< Block > &blocks)
- void paint (pro2::Window &window) const
- pro2::Rect collision_box () const
- int type () const
- pro2::Pt pos () const

5.8.1 Detailed Description

La classe Interactable engloba tot tipus d'objectes que tenen un comportament diferent amb els que el mario pot interactuar.

N'hi ha 3 tipus:

- 0: Moneda que és creada per un bloc quan el mario hi salta des de sota. Actua com una animació, pujant cap a munt durant un temps i desapareixent automàticament. La moneda s'afegeix al comptador de monedes
- 1: Bolet que fa que el mario es faci gran. Un cop apareix el bolet, aquest es mou i es xoca amb les parets fins que el mario el toca
- 2: La bandera del final del nivell. Un cop el mario la toca, es guanya el joc.
- 3: És un 'goomba' enemic. Si el mario hi salta a sobre mor i crea una moneda, si el 'goomba' toca al mario, el mario mor.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Interactable()

```
Interactable::Interactable (
            pro2::Pt pos,
            int type)  [inline]
```

5.8.3 Member Function Documentation

5.8.3.1 collision_box()

```
pro2::Rect Interactable::collision_box () const  [inline]
```

5.8.3.2 paint()

```
void Interactable::paint (
            pro2::Window & window) const
```

5.8.3.3 pos()

```
pro2::Pt Interactable::pos () const  [inline]
```

5.8.3.4 type()

```
int Interactable::type () const  [inline]
```

5.8.3.5 update()

```
bool Interactable::update (
            pro2::Window & window,
            const Finder< Platform > & platforms,
            const Finder< Block > & blocks)
```

The documentation for this class was generated from the following files:

- interactables.hh
- interactables.cc

5.9 Mario Class Reference

```
#include <mario.hh>
```

Public Member Functions

- Mario (int key_up, int key_left, int key_right, pro2::Pt pos)
- void paint (pro2::Window &window) const
- pro2::Rect collision_box () const
- pro2::Pt pos () const
- void set_y (int y)
- bool is_grounded () const
- void set_grounded (bool grounded)
- void toggle_grounded ()
- void jump ()
- void add_coin (int ammount=1)

  Afegeix monedes al comptador de monedes.
- int get_coin_count ()

  Retorna el valor del comptador de monedes.
- void update (pro2::Window &window, const Finder< Platform > &platforms, Finder< Block > &blocks, std↩ ::list< Interactable > &interactables)
- void set_state (int new_state)

  Canvia el estat del jugador a mario petit (0) o a mario gran (1). També hi ha els estats 2 i 3 per passar de petit a gran i viceversa (s'usen per a les animacions)
- int get_state () const

  Retorna l'estat del mario: mario petit (0) o mario gran (1)

5.9.1 Constructor & Destructor Documentation

5.9.1.1 Mario()

```
Mario::Mario (
            int key_up,
            int key_left,
            int key_right,
            pro2::Pt pos)  [inline]
```

5.9.2 Member Function Documentation

5.9.2.1 add_coin()

```
void Mario::add_coin (
            int ammount = 1)  [inline]
```

Afegeix monedes al comptador de monedes.

Parameters

ammount	Quantitat de monedes (per defecte: 1)

5.9.2.2 collision_box()

```
pro2::Rect Mario::collision_box () const
```

5.9.2.3 get_coin_count()

```
int Mario::get_coin_count ()  [inline]
```

Retorna el valor del comptador de monedes.

5.9.2.4 get_state()

```
int Mario::get_state () const  [inline]
```

Retorna l'estat del mario: mario petit (0) o mario gran (1)

5.9.2.5 is_grounded()

```
bool Mario::is_grounded () const  [inline]
```

5.9.2.6 jump()

```
void Mario::jump ()
```

5.9.2.7 paint()

```
void Mario::paint (
            pro2::Window & window) const
```

5.9.2.8 pos()

```
pro2::Pt Mario::pos () const  [inline]
```

5.9.2.9 set_grounded()

```
void Mario::set_grounded (
            bool grounded)  [inline]
```

5.9.2.10 set_state()

```
void Mario::set_state (
            int new_state)
```

Canvia el estat del jugador a mario petit (0) o a mario gran (1). També hi ha els estats 2 i 3 per passar de petit a gran i viceversa (s'usen per a les animacions)

Parameters

new_state	Nou estat al que passa el jugador

5.9.2.11 set_y()

```
void Mario::set_y (
            int y)  [inline]
```

5.9.2.12 toggle_grounded()

```
void Mario::toggle_grounded ()  [inline]
```

5.9.2.13 update()

```
void Mario::update (
            pro2::Window & window,
            const Finder< Platform > & platforms,
            Finder< Block > & blocks,
            std::list< Interactable > & interactables)
```

The documentation for this class was generated from the following files:

- mario.hh
- mario.cc

5.10 Platform Class Reference

`#include <platform.hh>`

Public Member Functions

- Platform ()
- Platform (const Platform &other)
- Platform (int left, int right, int top, int bottom)
- void paint (pro2::Window &window) const
- bool has_crossed_floor_downwards (pro2::Pt plast, pro2::Pt pcurr) const
- bool is_pt_inside (pro2::Pt pt) const
- int top () const
- pro2::Rect get_rect () const

5.10.1 Constructor & Destructor Documentation

5.10.1.1 Platform() [1/3]

```
Platform::Platform ()  [inline]
```

5.10.1.2 Platform() [2/3]

```
Platform::Platform (
            const Platform & other)  [inline]
```

5.10.1.3 Platform() [3/3]

```
Platform::Platform (
            int left,
            int right,
            int top,
            int bottom)  [inline]
```

5.10.2 Member Function Documentation

5.10.2.1 get_rect()

```
pro2::Rect Platform::get_rect () const  [inline]
```

5.10.2.2 has_crossed_floor_downwards()

```
bool Platform::has_crossed_floor_downwards (
            pro2::Pt plast,
            pro2::Pt pcurr) const
```

5.10.2.3 is_pt_inside()

```
bool Platform::is_pt_inside (
            pro2::Pt pt) const
```

5.10.2.4 paint()

```
void Platform::paint (
            pro2::Window & window) const
```

5.10.2.5 top()

```
int Platform::top () const  [inline]
```

The documentation for this class was generated from the following files:

- platform.hh
- platform.cc

5.11 pro2::Pt Struct Reference

```
#include <geometry.hh>
```

Public Member Functions

- Pt operator+ (const Pt &other) const
- Pt operator- (const Pt &other) const
- Pt & operator+= (const Pt &other)
- Pt & operator-= (const Pt &other)

Public Attributes

- int x = 0
- int y = 0

5.11.1 Member Function Documentation

5.11.1.1 operator+()

```
Pt pro2::Pt::operator+ (
            const Pt & other) const  [inline]
```

5.11.1.2 operator+=()

```
Pt & pro2::Pt::operator+= (
            const Pt & other)  [inline]
```

5.11.1.3 operator-()

```
Pt pro2::Pt::operator- (
            const Pt & other) const  [inline]
```

5.11.1.4 operator-=()

```
Pt & pro2::Pt::operator-= (
            const Pt & other)  [inline]
```

5.11.2 Member Data Documentation

5.11.2.1 x

```
int pro2::Pt::x = 0
```

5.11.2.2 y

```
int pro2::Pt::y = 0
```

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

- geometry.hh

5.12 pro2::Rect Struct Reference

```
#include <geometry.hh>
```

Public Member Functions

- int width () const
- int height () const
- Rect & operator+= (const Rect &other)
- Rect & operator-= (const Rect &other)

Public Attributes

- int left
- int top
- int right
- int bottom

5.12.1 Member Function Documentation

5.12.1.1 height()

```
int pro2::Rect::height () const  [inline]
```

5.12.1.2 operator+=()

```
Rect & pro2::Rect::operator+= (
            const Rect & other)  [inline]
```

5.12.1.3 operator-=()

```
Rect & pro2::Rect::operator-= (
            const Rect & other)  [inline]
```

5.12.1.4 width()

```
int pro2::Rect::width () const  [inline]
```

5.12.2 Member Data Documentation

5.12.2.1 bottom

```
int pro2::Rect::bottom
```

5.12.2.2 left

```
int pro2::Rect::left
```

5.12.2.3 right

```
int pro2::Rect::right
```

5.12.2.4 top

```
int pro2::Rect::top
```

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

- geometry.hh

5.13 StartScreen Class Reference

Aquesta classe s'encarrega de dibuixar el menú principal i gestionar la interacció de l'usuari.

```
#include <start_screen.hh>
```

Public Member Functions

- StartScreen (int width, int height, pro2::TextWriter TW)
- bool is_finished () const
- void process_keys (pro2::Window &window)
- void update (pro2::Window &window)
- void paint (pro2::Window &window)
- int exit_code () const

  Retorna 1 si s'ha pres el botó d'iniciar el joc i -1 altrament.
- void restart (pro2::Window &window)

  Restableix els valors per defecte.

5.13.1 Detailed Description

Aquesta classe s'encarrega de dibuixar el menú principal i gestionar la interacció de l'usuari.

```
Disposa de dos botons, un per iniciar el joc i l'altra per sortir.
Quan el mario mor, guanya o quan es prem la tecla "Escape" dins del joc, es retorna al menú principal.
```

5.13.2 Constructor & Destructor Documentation

5.13.2.1 StartScreen()

```
StartScreen::StartScreen (
            int width,
            int height,
            pro2::TextWriter TW)
```

5.13.3 Member Function Documentation

5.13.3.1 exit_code()

```
int StartScreen::exit_code () const  [inline]
```

Retorna 1 si s'ha pres el botó d'iniciar el joc i -1 altrament.

5.13.3.2 is_finished()

```
bool StartScreen::is_finished () const  [inline]
```

5.13.3.3 paint()

```
void StartScreen::paint (
            pro2::Window & window)
```

5.13.3.4 process_keys()

```
void StartScreen::process_keys (
            pro2::Window & window)
```

5.13.3.5 restart()

```
void StartScreen::restart (
            pro2::Window & window)
```

Restableix els valors per defecte.

5.13.3.6 update()

```
void StartScreen::update (
            pro2::Window & window)
```

The documentation for this class was generated from the following files:

- start_screen.hh
- start_screen.cc

5.14 pro2::TextWriter Class Reference

```
#include <text.hh>
```

Public Member Functions

- TextWriter (Font font, Palette palette)
- TextWriter (Font font, std::string palette_path)
- TextWriter (std::string font_path, Palette palette)
- TextWriter (std::string font_path, std::string palette_path)
- Sprite get_sprite (char ch) const

  Retorna el sprite del caràcter pintat amb la paleta. Si el caràcter no existeix a la font retorna un sprite amb el caràcter NULL (últim caràcter de la font)
- void set_font (Font font)
- void set_font (std::string path)
- const Font get_font ()
- void set_palette (Palette palette)
- void set_palette (std::string path)
- const Palette get_palette ()
- void set_charset (Charset charset)
- void set_charset (std::string path)
- const Charset get_charset ()
- void write_text (Window &window, const Pt &orig, const std::string &text, int space_between_chars=1, int size=4, Pt alignment={0, 0})

  Dibuixa un string com a text a la pantalla.

5.14.1 Detailed Description

S'encarrega d'emmagatzemar la font, paleta i charset carregats i dibuixar text a la pantalla

5.14.2 Constructor & Destructor Documentation

5.14.2.1 TextWriter() [1/4]

```
pro2::TextWriter::TextWriter (
            Font font,
            Palette palette)  [inline]
```

5.14.2.2 TextWriter() [2/4]

```
pro2::TextWriter::TextWriter (
            Font font,
            std::string palette_path)
```

5.14.2.3 TextWriter() [3/4]

```
pro2::TextWriter::TextWriter (
            std::string font_path,
            Palette palette)
```

5.14.2.4 TextWriter() [4/4]

```
pro2::TextWriter::TextWriter (
            std::string font_path,
            std::string palette_path)
```

5.14.3 Member Function Documentation

5.14.3.1 get_charset()

```
const Charset pro2::TextWriter::get_charset ()  [inline]
```

5.14.3.2 get_font()

```
const Font pro2::TextWriter::get_font ()  [inline]
```

5.14.3.3 get_palette()

```
const Palette pro2::TextWriter::get_palette ()  [inline]
```

5.14.3.4 get_sprite()

```
Sprite pro2::TextWriter::get_sprite (
            char ch) const
```

Retorna el sprite del caràcter pintat amb la paleta. Si el caràcter no existeix a la font retorna un sprite amb el caràcter NULL (últim caràcter de la font)

Parameters

ch	Caràcter

5.14.3.5 set_charset() [1/2]

```
void pro2::TextWriter::set_charset (
            Charset charset)  [inline]
```

5.14.3.6 set_charset() [2/2]

```
void pro2::TextWriter::set_charset (
            std::string path)
```

5.14.3.7 set_font() [1/2]

```
void pro2::TextWriter::set_font (
            Font font)  [inline]
```

5.14.3.8 set_font() [2/2]

```
void pro2::TextWriter::set_font (
            std::string path)
```

5.14.3.9 set_palette() [1/2]

```
void pro2::TextWriter::set_palette (
            Palette palette)  [inline]
```

5.14.3.10 set_palette() [2/2]

```
void pro2::TextWriter::set_palette (
            std::string path)
```

5.14.3.11 write_text()

```
void pro2::TextWriter::write_text (
            Window & window,
            const Pt & orig,
            const std::string & text,
            int space_between_chars = 1,
            int size = 4,
            Pt alignment = {0,0})
```

Dibuixa un string com a text a la pantalla.

Parameters

window	Finestra a on dibuixar
orig	Coordenades d'origen del text (coordenada esquerre superior)
text	String amb el text a dibuixar
space_between_chars	Espai que es deixarà entre caràcters
size	Tamany (gruix) del text
alignment	Determina com estarà alineat el text: {x_align, y_align} x_align i y_align poden tenir valors [0,1,2] que equivalen [left/top, centre, right/bottom]. Per exemple amb align {0,0}, tindrà a orig la cantonada esquerra superior del text

The documentation for this class was generated from the following files:

- text.hh
- text.cc

5.15 pro2::Window Class Reference

`#include <window.hh>`

Public Member Functions

- Window (std::string title, int width, int height, int zoom=1)

  Contruye una ventana con título, anchura y altura.
- ~Window ()

  Destruye una ventana, es decir, cierra la ventana abierta en el constructor.
- int width () const

  Devuelve el ancho de la ventana.
- int height () const

  Devuelve el alto de la ventana.
- bool next_frame ()

  Gestiona las tareas necesarias para pasar al siguiente fotograma.
- void clear (Color color=black)

  Rellena la ventana con un color.
- int frame_count () const

  Devuelve el contador de fotogramas pintados hasta el momento.
- bool is_key_down (int code) const

  Determina si cierta tecla estuvo presionada en el fotograma anterior.
- bool was_key_pressed (int code) const

  Determina si cierta tecla se presionó entre el fotograma anterior y el actual.
- bool is_modkey_down (ModKey key) const

  Determina si cierta tecla de control se presionó entre el fotograma anterior y el actual.
- bool is_mouse_down () const

  Determina si el botón izquierdo quedó en estado clicado en el fotograma anterior.
- bool was_mouse_pressed () const

  Determina si el botón izquierdo del ratón se clicó entre el fotograma anterior y el actual.
- Pt mouse_pos () const

  Devuelve la posición del cursor del ratón.

- void sleep (int ms) const

  Espera que pase un número `ms` de milisegundos sin hacer nada.
- Color get_pixel (Pt xy) const

  Obtiene el color de un pixel de la ventana.
- void set_pixel (Pt xy, Color color)

  Cambia un pixel de la ventana.
- void set_fps (int fps)

  Cambia los FPS de refresco de la ventana.
- void move_camera (Pt desplazamiento)

  Indica que la posición de la esquina superior izquierda de la ventana debería moverse según el vector `desplazamiento`.
- Pt camera_center () const

  Devuelve la posición del centro de la cámara.
- Rect camera_rect () const
- void set_camera_topleft (Pt topleft)

  Establece la posición de la esquina superior izquierda de la cámara.
- Pt topleft () const

  Devuelve la posición de la esquina superior izquierda de la cámara.

5.15.1 Detailed Description

La clase `Window` permite abrir ventanas en modo gráfico en Linux, MacOS y Windows. Tiene unos pocos métodos que permiten hacer programas simples que muestran gráficos, como pequeños juegos o editores.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 Window()

```
pro2::Window::Window (
            std::string title,
            int width,
            int height,
            int zoom = 1)
```

Contruye una ventana con título, anchura y altura.

El constructor abre una ventana, y el destructor la cierra.

El parámetro `zoom` permite visualizar con más comodidad contenido pixelado. Con `zoom = 1` cada pixel de la ventana se corresponde con un pixel de la pantalla. Con `zoom = 3`, cada píxel de la ventana se convierte en un cuadrado de 3x3 píxeles en la ventana.

Parameters

title	El título de la ventana (un literal de cadena de caracteres)
width	El ancho de la ventana en píxels.
height	El alto de la ventana en píxels.
zoom	El factor de aumento de cada píxel. (Es opcional, si no hay 4o parámetro toma valor 1)

5.15.2.2 ~Window()

```
pro2::Window::~Window ()  [inline]
```

Destruye una ventana, es decir, cierra la ventana abierta en el constructor.

5.15.3 Member Function Documentation

5.15.3.1 camera_center()

```
Pt pro2::Window::camera_center () const  [inline]
```

Devuelve la posición del centro de la cámara.

Returns

Un `Pt` con las coordenadas del centro de la cámara.

5.15.3.2 camera_rect()

```
Rect pro2::Window::camera_rect () const  [inline]
```

5.15.3.3 clear()

```
void pro2::Window::clear (
            Color color = black)
```

Rellena la ventana con un color.

Este método se puede llamar con un color o bien sin parámetros. Si se llama sin parámetros se toma el `color` por defecto, que es el negro (`black`). De lo contrario se usa el color indicado.

Parameters

color	El color a utilizar para pintar. Se puede usar uno de los valores del enumerado `Colors`, como `red`, o bien poner un entero en hexadecimal, como 0x0084fb, que equivale a los 3 valores RGB (o Red-Green-Blue) que conforman el color. Cualquier "color picker" de la web suele mostrar el color hexadecimal en la notación `#0084fb` (de CSS).

5.15.3.4 frame_count()

```
int pro2::Window::frame_count () const  [inline]
```

Devuelve el contador de fotogramas pintados hasta el momento.

Equivale a la cantidad de veces que se ha llamado a `next_frame`. Se incrementa en 1 unidad en cada fotograma.

Este valor es útil al hacer animaciones, ya que permite saber, de una secuencia de imágenes, cuál habría que usar en cada momento.

Returns

Un entero que corresponde al contador de fotogramas mostrados desde que la ventana se creó.

5.15.3.5 get_pixel()

```
Color pro2::Window::get_pixel (
            Pt xy) const  [inline]
```

Obtiene el color de un pixel de la ventana.

Parameters

xy	Coordenadas del pixel de la pantalla del que se quiere saber el color.

Returns

El color del pixel en las coordenadas indicadas.

5.15.3.6 height()

```
int pro2::Window::height () const  [inline]
```

Devuelve el alto de la ventana.

5.15.3.7 is_key_down()

```
bool pro2::Window::is_key_down (
            int code) const  [inline]
```

Determina si cierta tecla estuvo presionada en el fotograma anterior.

El método `next_frame` recoge todas los eventos de teclado y ratón que han ocurrido desde la llamada anterior a `next_frame` (o desde la creación de la ventana) y mantiene el estado de todas las teclas y botones del ratón fijo durante el fotograma actual. Así pues, el método `is_key_down` simplemente consulta ese estado, que se mantiene fijo hasta la siguiente llamada a `next_frame`.

Ejemplo:

```
if (window.is_key_down('S')) { ... }
if (window.is_key_down('1')) { ... }
if (window.is_key_down(Key::Escape)) { ... }
```

Parameters

code	El código de la tecla de la que se quiere saber si estaba presionada. El código de la letra es, o bien el código ASCII de la letra mayúscula correspondiente, el código ASCII del dígito correspondiente, o bien uno de los valores del `enum Key`, que recoge las teclas más típicas, incluyendo flechas, return, esc, tab, etc.

Returns

`true` cuando la tecla `code` estaba presionada al empezar el fotograma actual.

5.15.3.8 is_modkey_down()

```
bool pro2::Window::is_modkey_down (
            ModKey key) const  [inline]
```

Determina si cierta tecla de control se presionó entre el fotograma anterior y el actual.

Método análogo a `is_key_down` pero para las teclas de control siguientes: Ctrl, Alt, Shift y Meta. Hay un enumerado de nombre `ModKey` con las 4 teclas: `ModKey::Ctrl`, `ModKey::Alt`, `ModKey::Shift`, y `ModKey::Meta`.

Parameters

key	La tecla de la que se quiere consultar el estado.

Returns

`true` si el estado de la tecla era "presionado" al entrar al fotograma actual.

5.15.3.9 is_mouse_down()

```
bool pro2::Window::is_mouse_down () const  [inline]
```

Determina si el botón izquierdo quedó en estado clicado en el fotograma anterior.

Este método se comporta como `is_key_down`, consulta la documentación de `is_key_down` para saber cómo opera.

Returns

`true` si el botón del ratón quedó clicado al final del fotograma actual.

5.15.3.10 mouse_pos()

```
Pt pro2::Window::mouse_pos () const
```

Devuelve la posición del cursor del ratón.

Returns

Una tupla de tipo `Pt`, con campos `x` e `y`, que se corresponden con las coordenadas de la posición del ratón.

5.15.3.11 move_camera()

```
void pro2::Window::move_camera (
            Pt desplazamiento)  [inline]
```

Indica que la posición de la esquina superior izquierda de la ventana debería moverse según el vector `desplazamiento`.

La cámara no se mueve instantáneamente, sino que se desplaza a la nueva posición a una velocidad constante.

Parameters

desplazamiento	Vector de desplazamiento

5.15.3.12 next_frame()

```
bool pro2::Window::next_frame ()
```

Gestiona las tareas necesarias para pasar al siguiente fotograma.

En todo programa gráfico es necesario: 1) pintar en una superfície, típicamente en memoria, 2) transferir lo que se ha pintado a la pantalla, 3) procesar eventos ocurridos como presión de teclas o movimiento del ratón y actualizar su estado, y 4) esperar el tiempo que quede hasta el siguiente fotograma (en función de la velocidad de refresco, que suele ser de 60Hz, lo que equivale a 16ms por fotograma).

`next_frame` hace todas estas cosas en una sola llamada. Además devuelve `false` cuando se ha clicado el botón de cerrar la ventana (típicamente arriba a la derecha, y con una "x"), de forma que se pueda saber si se debe continuar en un bucle de pintado de fotogramas.

El uso típico es el siguiente:

```
while (window.next_frame()) {
    // usar los métodos de detección de teclas o ratón, y set_pixel para pintar...
}
```

Es decir, hasta que no se cierre la ventana llamamos métodos de la ventana para hacer operaciones que resulten en el pintado de la ventana de cierta manera y `next_frame` se hace cargo del resto.

Con respecto al teclado y ratón, `next_frame` recoge todos los eventos (presión y soltado de teclas, clicks y movimiento del ratón) que han ocurrido entre el fotograma anterior y el actual, y con todos ellos actualiza el estado final de cada tecla, botón del ratón y posición. Así pues, el usuario de la clase `Window` tiene acceso al estado exacto de las teclas y el ratón en el instante en que se pasa al fotograma actual, y ese estado se conserva fijo mientras transcurre el tiempo entre el fotograma actual y el siguiente, en el que `next_frame` vuelve a revisar los eventos ocurridos en ese intervalo de tiempo.

Returns

`true` si el programa debe seguir (NO se ha clicado el botón de cerrar la ventana), `false` en caso contrario.

5.15.3.13 set_camera_topleft()

```
void pro2::Window::set_camera_topleft (
            Pt topleft)  [inline]
```

Establece la posición de la esquina superior izquierda de la cámara.

Este método mueve la cámara instantáneamente a la nueva posición.Alt

Parameters

topleft	La nueva posición absoluta de la cámara, que se aplica instantáneamente.

5.15.3.14 set_fps()

```
void pro2::Window::set_fps (
            int fps)  [inline]
```

Cambia los FPS de refresco de la ventana.

En función de la velocidad de refresco de la pantalla que queramos, el tiempo a esperar entre que pintamos un fotograma y el siguiente puede variar. Este método calcula un tiempo de espera entre una llamada a `next_frame` y la siguiente, para que se produzca exactamente un número de fotogramas por segundo.

Parameters

fps	Número de fotogramas por segundo que se quieren mostrar.

Precondition

`fps` > 0 && `fps` < 240.

5.15.3.15 set_pixel()

```
void pro2::Window::set_pixel (
            Pt xy,
            Color color)
```

Cambia un pixel de la ventana.

En realidad, `set_pixel` no cambia la ventana directamente, sinó un "buffer" interno que se vuelca en la pantalla de golpe en el momento de llamar a `next_frame`. Esto es más eficiente y maximiza el tiempo en que el fotograma está inmóvil en la pantalla mostrando una imagen fija, ya que el pintado podría llevar tanto tiempo que los fotogramas no se verían completos en la pantalla durante los 16ms (a 60Hz) en que deben estar visibles.

Parameters

xy	Coordenadas del pixel que se quiere cambiar
color	Color que se quiere poner en el pixel indicado

5.15.3.16 sleep()

```
void pro2::Window::sleep (
            int ms) const  [inline]
```

Espera que pase un número `ms` de milisegundos sin hacer nada.

En ese intervalo de tiempo el programa estará esperando que el método vuelva de la llamada, y por tanto no se ejecutará ninguna instrucción.

Parameters

ms	Número de milisegundos a esperar.

5.15.3.17 topleft()

```
Pt pro2::Window::topleft () const  [inline]
```

Devuelve la posición de la esquina superior izquierda de la cámara.

Returns

Un `Pt` con las coordenadas de la esquina superior izquierda de la cámara.

5.15.3.18 was_key_pressed()

```
bool pro2::Window::was_key_pressed (
            int code) const  [inline]
```

Determina si cierta tecla se presionó entre el fotograma anterior y el actual.

(En el método `is_key_down` se explica mejor el funcionamiento de los eventos.)

Ejemplo:

```
if (window.was_key_pressed('S')) { ... }
if (window.was_key_pressed('1')) { ... }
if (window.was_key_pressed(Key::Escape)) { ... }
```

Parameters

code	El código de la tecla de la que se quiere saber si estaba presionada. El código de la letra es, o bien el código ASCII de la letra mayúscula correspondiente, el código ASCII del dígito correspondiente, o bien uno de los valores del `enum Key`, que recoge las teclas más típicas, incluyendo flechas, return, esc, tab, etc.

Returns

`true` cuando la tecla `code` estaba presionada al empezar el fotograma actual.

5.15.3.19 was_mouse_pressed()

```
bool pro2::Window::was_mouse_pressed () const  [inline]
```

Determina si el botón izquierdo del ratón se clicó entre el fotograma anterior y el actual.

Este método se comporta como `was_key_pressed`, consulta la documentación de `was_key_pressed` para saber cómo opera.

Returns

`true` si el botón del ratón se clicó entre el fotograma anterior y el actual.

5.15.3.20 width()

```
int pro2::Window::width () const  [inline]
```

Devuelve el ancho de la ventana.

The documentation for this class was generated from the following files:

- window.hh
- window.cc

Chapter 6

File Documentation

6.1 block.cc File Reference

```
#include "block.hh"
```

6.2 block.hh File Reference

```
#include <vector>
#include <iostream>
#include "window.hh"
#include "geometry.hh"
#include "utils.hh"
```

Classes

- class Block

6.3 block.hh

Go to the documentation of this file.

```
00001 #ifndef BLOCK_HH
00002 #define BLOCK_HH
00003
00004 #include <vector>
00005 #include <iostream>
00006 #include "window.hh"
00007 #include "geometry.hh"
00008 #include "utils.hh"
00009
00010
00011 class Block {
00012  private:
00013     pro2::Pt pos_;
00014
00015     int block_type_; // 0 si és tipus totxo, 1 interrogant, 2 activat
00016     int has_object_; // 1 si té una moneda, 2 si té un bolet
00017
```

```
00018     static const std::vector<std::vector<int» platform_texture_;
00019
00020  public:
00021     Block(pro2::Pt pos = {0,0}, int type = 0, int has_object = 0) : pos_(pos), block_type_(type),
      has_object_(has_object) {}
00022
00023     Block(const Block& other)
00024         : pos_(other.pos_), block_type_(other.block_type_) {}
00025
00026     void paint(pro2::Window& window, int anim_frame) const;
00027
00028     pro2::Pt pos() const {return pos_;}
00029
00040     int block_type() const {return block_type_;}
00041
00042     inline pro2::Rect get_rect() const {
00043         return {pos_.x, pos_.y, pos_.x + sz_w, pos_.y + sz_h};
00044     }
00045
00049     inline std::vector<std::vector<int» get_sprite(int anim_frame) const {
00050         switch (block_type_)
00051         {
00052         case 0:
00053             return sprites[0];
00054         case 1:
00055             return sprites[animation[anim_frame]];
00056         case 2:
00057             return sprites[1];
00058         case 3:
00059             return sprites[5];
00060         default:
00061             return sprites[0];
00062         }
00063     }
00064
00076     int check_bumped(int state) {
00077         if (block_type_ == 0) {
00078             if (has_object_ == 1 and state == 1) {
00079                 block_type_ = 2;
00080                 return 2;
00081             }
00082             else {
00083                 if (state == 1) return 1;
00084                 else {
00085                     return 0;
00086                 }
00087             }
00088         }
00089         else if (block_type_ == 1) {
00090             block_type_ = 2;
00091             if (has_object_ == 1) return 2;
00092             else if (has_object_ == 2) return 3;
00093             else return 0;
00094         }
00095         else return 0;
00096     }
00097
00098     static const std::vector<std::vector<std::vector<int»» sprites;
00099  private:
00100     static const int sz_h;
00101     static const int sz_w;
00102
00103     static const std::vector<int> animation;
00104 };
00105
00106
00107 #endif
```

6.4 coin.cc File Reference

```
#include "coin.hh"
#include "utils.hh"
```

6.5 coin.hh File Reference

```
#include <iostream>
#include <set>
#include <vector>
#include "platform.hh"
#include "window.hh"
```

Classes

- class Coin

6.6 coin.hh

Go to the documentation of this file.

```
00001 #ifndef COIN_HH
00002 #define COIN_HH
00003
00004 #include <iostream>
00005 #include <set>
00006 #include <vector>
00007 #include "platform.hh"
00008 #include "window.hh"
00009
00010 class Coin {
00011  private:
00012     double gravity = 1;
00013
00014     pro2::Pt pos_, last_pos_;    // Pos és la posició de la cantonada superior esquerra
00015     pro2::DoubPt speed_;
00016     pro2::DoubPt accel_;         // L'acceleració per defecte és -gravetat (moneda estàtica)
00017     pro2::DoubPt drag_coef_;     // F = -b*v
00018
00019     bool grounded_ = false;
00020
00021     void apply_physics_();
00022
00023  public:
00024
00034     Coin(pro2::Pt pos, pro2::DoubPt speed = {0, 0}, pro2::DoubPt accel = {0, -1}, pro2::DoubPt drag =
      {0.075, 0.075}) :
00035         pos_(pos), last_pos_(pos), speed_(speed), accel_(accel), drag_coef_(drag) {}
00036
00042     void paint(pro2::Window& window, int anim_frame) const;
00043
00044     pro2::Pt pos() const {
00045         return pos_;
00046     }
00047
00048     inline pro2::Rect get_rect() const {
00049         return {pos_.x - sz_w/2, pos_.y - sz_h, pos_.x + sz_w/2, pos_.y};
00050     }
00051
00055     inline std::vector<std::vector<int>> get_sprite(int anim_frame) const {
00056         return sprites[animation[anim_frame]];
00057     }
00058
00059     void set_y(int y) {
00060         pos_.y = y;
00061     }
00062
00063     bool is_grounded() const {
00064         return grounded_;
00065     }
00066
00067     void set_grounded(bool grounded) {
00068         grounded_ = grounded;
00069         if (grounded_) {
00070             speed_.y = 0;
00071         }
00072     }
```

```
00073
00074     void toggle_grounded() {
00075         set_grounded(!grounded_);
00076     }
00077
00078     void update(pro2::Window& window, const std::set<Platform *>& platforms);
00079
00080     static const std::vector<std::vector<std::vector<int>> sprites;    // Vector de els diferents
      sprites de l'animació
00081  private:
00082     static const std::vector<int> animation;                           // Vector amb els índex dels
      sprites de l'animació
00083     static const int sz_h;
00084     static const int sz_w;
00085 };
00086
00087 #endif
```

6.7 fenster.h File Reference

```
#include <X11/XKBlib.h>
#include <X11/Xlib.h>
#include <X11/keysym.h>
#include <time.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

Classes

- struct fenster

Macros

- #define _DEFAULT_SOURCE 1
- #define FENSTER_API extern
- #define fenster_pixel(f, x, y)

Functions

- FENSTER_API int fenster_open (struct fenster *f)
- FENSTER_API int fenster_loop (struct fenster *f)
- FENSTER_API void fenster_close (struct fenster *f)
- FENSTER_API void fenster_sleep (int64_t ms)
- FENSTER_API int64_t fenster_time (void)

6.7.1 Macro Definition Documentation

6.7.1.1 _DEFAULT_SOURCE

```
#define _DEFAULT_SOURCE 1
```

6.7.1.2 FENSTER_API

```
#define FENSTER_API extern
```

6.7.1.3 fenster_pixel

```
#define fenster_pixel(
              f,
              x,
              y)
```

Value:

```
((f)->buf[((y) * (f)->width) + (x)])
```

6.7.2 Function Documentation

6.7.2.1 fenster_close()

```
FENSTER_API void fenster_close (
             struct fenster * f)
```

6.7.2.2 fenster_loop()

```
FENSTER_API int fenster_loop (
             struct fenster * f)
```

6.7.2.3 fenster_open()

```
FENSTER_API int fenster_open (
             struct fenster * f)
```

6.7.2.4 fenster_sleep()

```
FENSTER_API void fenster_sleep (
             int64_t ms)
```

6.7.2.5 fenster_time()

```
FENSTER_API int64_t fenster_time (
             void )
```

6.8 fenster.h

Go to the documentation of this file.

```
00001 #ifndef FENSTER_H
00002 #define FENSTER_H
00003
00004 #if defined(__APPLE__)
00005 #include <CoreGraphics/CoreGraphics.h>
00006 #include <objc/NSObjCRuntime.h>
00007 #include <objc/objc-runtime.h>
00008 #elif defined(_WIN32)
00009 #include <windows.h>
00010 #else
00011 #define _DEFAULT_SOURCE 1
00012 #include <X11/XKBlib.h>
00013 #include <X11/Xlib.h>
00014 #include <X11/keysym.h>
00015 #include <time.h>
00016 #endif
00017
00018 #include <stdint.h>
00019 #include <stdlib.h>
00020 #include <string.h>
00021
00022 struct fenster {
00023     const char *title;
00024     const int   width;
00025     const int   height;
00026     uint32_t   *buf;
00027     int         keys[256]; /* keys are mostly ASCII, but arrows are 17..20 */
00028     int         mod;       /* mod is 4 bits mask, ctrl=1, shift=2, alt=4, meta=8 */
00029     int         x;
00030     int         y;
00031     int         mouse;
00032 #if defined(__APPLE__)
00033     id wnd;
00034 #elif defined(_WIN32)
00035     HWND hwnd;
00036 #else
00037     Display *dpy;
00038     Window   w;
00039     GC       gc;
00040     XImage  *img;
00041 #endif
00042 };
00043
00044 #ifndef FENSTER_API
00045 #define FENSTER_API extern
00046 #endif
00047 FENSTER_API int     fenster_open(struct fenster *f);
00048 FENSTER_API int     fenster_loop(struct fenster *f);
00049 FENSTER_API void    fenster_close(struct fenster *f);
00050 FENSTER_API void    fenster_sleep(int64_t ms);
00051 FENSTER_API int64_t fenster_time(void);
00052 #define fenster_pixel(f, x, y) ((f)->buf[((y) * (f)->width) + (x)])
00053
00054 #ifndef FENSTER_HEADER
00055 #if defined(__APPLE__)
00056 #define msg(r, o, s) ((r(*)(id, SEL))objc_msgSend)(o, sel_getUid(s))
00057 #define msg1(r, o, s, A, a) ((r(*)(id, SEL, A))objc_msgSend)(o, sel_getUid(s), a)
00058 #define msg2(r, o, s, A, a, B, b) ((r(*)(id, SEL, A, B))objc_msgSend)(o, sel_getUid(s), a, b)
00059 #define msg3(r, o, s, A, a, B, b, C, c) \
00060     ((r(*)(id, SEL, A, B, C))objc_msgSend)(o, sel_getUid(s), a, b, c)
00061 #define msg4(r, o, s, A, a, B, b, C, c, D, d) \
00062     ((r(*)(id, SEL, A, B, C, D))objc_msgSend)(o, sel_getUid(s), a, b, c, d)
00063
00064 #define cls(x) ((id)objc_getClass(x))
00065
00066 extern id const NSDefaultRunLoopMode;
00067 extern id const NSApp;
00068
00069 static void fenster_draw_rect(id v, SEL s, CGRect r) {
00070     (void)r, (void)s;
00071     struct fenster *f = (struct fenster *)objc_getAssociatedObject(v, "fenster");
00072     CGContextRef    context =
00073         msg(CGContextRef, msg(id, cls("NSGraphicsContext"), "currentContext"), "graphicsPort");
00074     CGColorSpaceRef   space = CGColorSpaceCreateDeviceRGB();
00075     CGDataProviderRef provider =
00076         CGDataProviderCreateWithData(NULL, f->buf, f->width * f->height * 4, NULL);
00077     CGImageRef img = CGImageCreate(f->width, f->height, 8, 32, f->width * 4, space,
00078                                    kCGImageAlphaNoneSkipFirst | kCGBitmapByteOrder32Little,
00079                                    provider, NULL, false, kCGRenderingIntentDefault);
00080     CGColorSpaceRelease(space);
00081     CGDataProviderRelease(provider);
00082     CGContextDrawImage(context, CGRectMake(0, 0, f->width, f->height), img);
```

```
00083     CGImageRelease(img);
00084 }
00085
00086 static BOOL fenster_should_close(id v, SEL s, id w) {
00087     (void)v, (void)s, (void)w;
00088     msg1(void, NSApp, "terminate:", id, NSApp);
00089     return YES;
00090 }
00091
00092 FENSTER_API int fenster_open(struct fenster *f) {
00093     msg(id, cls("NSApplication"), "sharedApplication");
00094     msg1(void, NSApp, "setActivationPolicy:", NSInteger, 0);
00095     f->wnd = msg4(id, msg(id, cls("NSWindow"), "alloc"),
00096                   "initWithContentRect:styleMask:backing:defer:", CGRect,
00097                   CGRectMake(0, 0, f->width, f->height), NSUInteger, 3, NSUInteger, 2, BOOL, NO);
00098     Class windelegate = objc_allocateClassPair((Class)cls("NSObject"), "FensterDelegate", 0);
00099     class_addMethod(windelegate, sel_getUid("windowShouldClose:"), (IMP)fenster_should_close,
00100                     "c@:@");
00101     objc_registerClassPair(windelegate);
00102     msg1(void, f->wnd, "setDelegate:", id, msg(id, msg(id, (id)windelegate, "alloc"), "init"));
00103     Class c = objc_allocateClassPair((Class)cls("NSView"), "FensterView", 0);
00104     class_addMethod(c, sel_getUid("drawRect:"), (IMP)fenster_draw_rect, "i@:@@");
00105     objc_registerClassPair(c);
00106
00107     id v = msg(id, msg(id, (id)c, "alloc"), "init");
00108     msg1(void, f->wnd, "setContentView:", id, v);
00109     objc_setAssociatedObject(v, "fenster", (id)f, OBJC_ASSOCIATION_ASSIGN);
00110
00111     id title = msg1(id, cls("NSString"), "stringWithUTF8String:", const char *, f->title);
00112     msg1(void, f->wnd, "setTitle:", id, title);
00113     msg1(void, f->wnd, "makeKeyAndOrderFront:", id, nil);
00114     msg(void, f->wnd, "center");
00115     msg1(void, NSApp, "activateIgnoringOtherApps:", BOOL, YES);
00116     return 0;
00117 }
00118
00119 FENSTER_API void fenster_close(struct fenster *f) {
00120     msg(void, f->wnd, "close");
00121 }
00122
00123 // clang-format off
00124 static const uint8_t FENSTER_KEYCODES[128] =
      {65,83,68,70,72,71,90,88,67,86,0,66,81,87,69,82,89,84,49,50,51,52,54,53,61,57,55,45,56,48,93,79,85,91,73,80,10,76,74,39,
00125 // clang-format on
00126
00127 FENSTER_API int fenster_loop(struct fenster *f) {
00128     msg1(void, msg(id, f->wnd, "contentView"), "setNeedsDisplay:", BOOL, YES);
00129     id ev = msg4(id, NSApp, "nextEventMatchingMask:untilDate:inMode:dequeue:", NSUInteger,
00130                  NSUIntegerMax, id, NULL, id, NSDefaultRunLoopMode, BOOL, YES);
00131     if (!ev) {
00132         return 0;
00133     }
00134     NSUInteger evtype = msg(NSUInteger, ev, "type");
00135     switch (evtype) {
00136         case 1: /* NSEventTypeMouseDown */
00137             f->mouse |= 1;
00138             break;
00139         case 2: /* NSEventTypeMouseUp*/
00140             f->mouse &= ~1;
00141             break;
00142         case 5:
00143         case 6: { /* NSEventTypeMouseMoved */
00144             CGPoint xy = msg(CGPoint, ev, "locationInWindow");
00145             f->x = (int)xy.x;
00146             f->y = (int)(f->height - xy.y);
00147             return 0;
00148         }
00149         case 10: /*NSEventTypeKeyDown*/
00150         case 11: /*NSEventTypeKeyUp:*/ {
00151             NSUInteger k = msg(NSUInteger, ev, "keyCode");
00152             f->keys[k < 127 ? FENSTER_KEYCODES[k] : 0] = evtype == 10;
00153             NSUInteger mod = msg(NSUInteger, ev, "modifierFlags") » 17;
00154             f->mod = (mod & 0xc) | ((mod & 1) « 1) | ((mod » 1) & 1);
00155             return 0;
00156         }
00157     }
00158     msg1(void, NSApp, "sendEvent:", id, ev);
00159     return 0;
00160 }
00161 #elif defined(_WIN32)
00162 // clang-format off
00163 static const uint8_t FENSTER_KEYCODES[] =
      {0,27,49,50,51,52,53,54,55,56,57,48,45,61,8,9,81,87,69,82,84,89,85,73,79,80,91,93,10,0,65,83,68,70,71,72,74,75,76,59,39,
00164 // clang-format on
00165
00166 typedef struct BINFO {
00167     BITMAPINFOHEADER bmiHeader;
```

```
00168     RGBQUAD          bmiColors[3];
00169 } BINFO;
00170
00171 static LRESULT CALLBACK fenster_wndproc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {
00172     struct fenster *f = (struct fenster *)GetWindowLongPtr(hwnd, GWLP_USERDATA);
00173     switch (msg) {
00174         case WM_PAINT: {
00175             PAINTSTRUCT ps;
00176             HDC         hdc = BeginPaint(hwnd, &ps);
00177             HDC         memdc = CreateCompatibleDC(hdc);
00178             HBITMAP     hbmp = CreateCompatibleBitmap(hdc, f->width, f->height);
00179             HBITMAP     oldbmp = (HBITMAP)SelectObject(memdc, hbmp);
00180             BINFO       bi = {{sizeof(bi), f->width, -f->height, 1, 32, BI_BITFIELDS}};
00181             bi.bmiColors[0].rgbRed = 0xff;
00182             bi.bmiColors[1].rgbGreen = 0xff;
00183             bi.bmiColors[2].rgbBlue = 0xff;
00184             SetDIBitsToDevice(memdc, 0, 0, f->width, f->height, 0, 0, 0, f->height, f->buf,
00185                               (BITMAPINFO *)&bi, DIB_RGB_COLORS);
00186             BitBlt(hdc, 0, 0, f->width, f->height, memdc, 0, 0, SRCCOPY);
00187             SelectObject(memdc, oldbmp);
00188             DeleteObject(hbmp);
00189             DeleteDC(memdc);
00190             EndPaint(hwnd, &ps);
00191         } break;
00192         case WM_CLOSE:
00193             DestroyWindow(hwnd);
00194             break;
00195         case WM_LBUTTONDOWN:
00196         case WM_LBUTTONUP:
00197             f->mouse = (msg == WM_LBUTTONDOWN);
00198             break;
00199         case WM_MOUSEMOVE:
00200             f->y = HIWORD(lParam), f->x = LOWORD(lParam);
00201             break;
00202         case WM_KEYDOWN:
00203         case WM_KEYUP: {
00204             f->mod = ((GetKeyState(VK_CONTROL) & 0x8000) >> 15) |
00205                      ((GetKeyState(VK_SHIFT) & 0x8000) >> 14) |
00206                      ((GetKeyState(VK_MENU) & 0x8000) >> 13) |
00207                      (((GetKeyState(VK_LWIN) | GetKeyState(VK_RWIN)) & 0x8000) >> 12);
00208             f->keys[FENSTER_KEYCODES[HIWORD(lParam) & 0x1ff]] = !((lParam >> 31) & 1);
00209         } break;
00210         case WM_DESTROY:
00211             PostQuitMessage(0);
00212             break;
00213         default:
00214             return DefWindowProc(hwnd, msg, wParam, lParam);
00215     }
00216     return 0;
00217 }
00218
00219 FENSTER_API int fenster_open(struct fenster *f) {
00220     HINSTANCE  hInstance = GetModuleHandle(NULL);
00221     WNDCLASSEX wc = {0};
00222     wc.cbSize = sizeof(WNDCLASSEX);
00223     wc.style = CS_VREDRAW | CS_HREDRAW;
00224     wc.lpfnWndProc = fenster_wndproc;
00225     wc.hInstance = hInstance;
00226     wc.lpszClassName = f->title;
00227     RegisterClassEx(&wc);
00228     f->hwnd =
00229         CreateWindowEx(WS_EX_CLIENTEDGE, f->title, f->title, WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
00230                        CW_USEDEFAULT, f->width, f->height, NULL, NULL, hInstance, NULL);
00231
00232     if (f->hwnd == NULL) {
00233         return -1;
00234     }
00235     SetWindowLongPtr(f->hwnd, GWLP_USERDATA, (LONG_PTR)f);
00236     ShowWindow(f->hwnd, SW_NORMAL);
00237     UpdateWindow(f->hwnd);
00238     return 0;
00239 }
00240
00241 FENSTER_API void fenster_close(struct fenster *f) {
00242     (void)f;
00243 }
00244
00245 FENSTER_API int fenster_loop(struct fenster *f) {
00246     MSG msg;
00247     while (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {
00248         if (msg.message == WM_QUIT) {
00249             return -1;
00250         }
00251         TranslateMessage(&msg);
00252         DispatchMessage(&msg);
00253     }
00254     InvalidateRect(f->hwnd, NULL, TRUE);
```

```
00255     return 0;
00256 }
00257 #else
00258 // clang-format off
00259 static int FENSTER_KEYCODES[124] =
      {XK_BackSpace,8,XK_Delete,127,XK_Down,18,XK_End,5,XK_Escape,27,XK_Home,2,XK_Insert,26,XK_Left,20,XK_Page_Down,4,XK_Page_
00260 // clang-format on
00261
00262 FENSTER_API int fenster_open(struct fenster *f) {
00263     f->dpy = XOpenDisplay(NULL);
00264     int screen = DefaultScreen(f->dpy);
00265     f->w = XCreateSimpleWindow(f->dpy, RootWindow(f->dpy, screen), 0, 0, f->width, f->height, 0,
00266                                BlackPixel(f->dpy, screen), WhitePixel(f->dpy, screen));
00267     f->gc = XCreateGC(f->dpy, f->w, 0, 0);
00268     XSelectInput(f->dpy, f->w,
00269                  ExposureMask | KeyPressMask | KeyReleaseMask | ButtonPressMask |
00270                      ButtonReleaseMask | PointerMotionMask);
00271     XStoreName(f->dpy, f->w, f->title);
00272     XMapWindow(f->dpy, f->w);
00273     XSync(f->dpy, f->w);
00274     f->img = XCreateImage(f->dpy, DefaultVisual(f->dpy, 0), 24, ZPixmap, 0, (char *)f->buf,
00275                           f->width, f->height, 32, 0);
00276
00277     Atom wmDelete = XInternAtom(f->dpy, "WM_DELETE_WINDOW", True);
00278     XSetWMProtocols(f->dpy, f->w, &wmDelete, 1);
00279     return 0;
00280 }
00281
00282 FENSTER_API void fenster_close(struct fenster *f) {
00283     XCloseDisplay(f->dpy);
00284 }
00285
00286 FENSTER_API int fenster_loop(struct fenster *f) {
00287     XEvent ev;
00288     XPutImage(f->dpy, f->w, f->gc, f->img, 0, 0, 0, 0, f->width, f->height);
00289     XFlush(f->dpy);
00290     while (XPending(f->dpy)) {
00291         XNextEvent(f->dpy, &ev);
00292         switch (ev.type) {
00293             case ButtonPress:
00294             case ButtonRelease:
00295                 f->mouse = (ev.type == ButtonPress);
00296                 break;
00297             case MotionNotify:
00298                 f->x = ev.xmotion.x, f->y = ev.xmotion.y;
00299                 break;
00300             case KeyPress:
00301             case KeyRelease: {
00302                 int m = ev.xkey.state;
00303                 int k = XkbKeycodeToKeysym(f->dpy, ev.xkey.keycode, 0, 0);
00304                 for (unsigned int i = 0; i < 124; i += 2) {
00305                     if (FENSTER_KEYCODES[i] == k) {
00306                         f->keys[FENSTER_KEYCODES[i + 1]] = (ev.type == KeyPress);
00307                         break;
00308                     }
00309                 }
00310                 f->mod = (!!(m & ControlMask)) | (!!(m & ShiftMask) << 1) |
00311                          (!!(m & Mod1Mask) << 2) | (!!(m & Mod4Mask) << 3);
00312                 break;
00313             }
00314             case ClientMessage:
00315                 return -1;
00316         }
00317     }
00318     return 0;
00319 }
00320 #endif
00321
00322 #ifdef _WIN32
00323 FENSTER_API void fenster_sleep(int64_t ms) {
00324     Sleep(ms);
00325 }
00326
00327 FENSTER_API int64_t fenster_time() {
00328     LARGE_INTEGER freq, count;
00329     QueryPerformanceFrequency(&freq);
00330     QueryPerformanceCounter(&count);
00331     return (int64_t)(count.QuadPart * 1000.0 / freq.QuadPart);
00332 }
00333 #else
00334 FENSTER_API void fenster_sleep(int64_t ms) {
00335     struct timespec ts;
00336     ts.tv_sec = ms / 1000;
00337     ts.tv_nsec = (ms % 1000) * 1000000;
00338     nanosleep(&ts, NULL);
00339 }
00340
```

```
00341 FENSTER_API int64_t fenster_time(void) {
00342     struct timespec time;
00343     clock_gettime(CLOCK_REALTIME, &time);
00344     return time.tv_sec * 1000 + (time.tv_nsec / 1000000);
00345 }
00346 #endif
00347
00348 #ifdef __cplusplus
00349 class Fenster {
00350     struct fenster f;
00351     int64_t        now;
00352
00353   public:
00354     Fenster(const int w, const int h, const char *title)
00355         : f{.title = title, .width = w, .height = h} {
00356         this->f.buf = new uint32_t[w * h];
00357         this->now = fenster_time();
00358         fenster_open(&this->f);
00359     }
00360
00361     ~Fenster() {
00362         fenster_close(&this->f);
00363         delete[] this->f.buf;
00364     }
00365
00366     bool loop(const int fps) {
00367         int64_t t = fenster_time();
00368         if (t - this->now < 1000 / fps) {
00369             fenster_sleep(t - now);
00370         }
00371         this->now = t;
00372         return fenster_loop(&this->f) == 0;
00373     }
00374
00375     inline uint32_t& px(const int x, const int y) { return fenster_pixel(&this->f, x, y); }
00376
00377     bool key(int c) { return c >= 0 && c < 128 ? this->f.keys[c] : false; }
00378
00379     int x() { return this->f.x; }
00380
00381     int y() { return this->f.y; }
00382
00383     int mouse() { return this->f.mouse; }
00384
00385     int mod() { return this->f.mod; }
00386 };
00387 #endif /* __cplusplus */
00388
00389 #endif /* !FENSTER_HEADER */
00390 #endif /* FENSTER_H */
```

6.9 finder.hh File Reference

```
#include "geometry.hh"
#include <algorithm>
#include <map>
#include <set>
#include <list>
#include <iostream>
```

Classes

- class Finder< T >

Variables

- const int NUM_DIVS = 32
- const int MAX_SZ = 20000
- const int DIVIDER = MAX_SZ/NUM_DIVS

6.9.1 Variable Documentation

6.9.1.1 DIVIDER

```
const int DIVIDER = MAX_SZ/NUM_DIVS
```

6.9.1.2 MAX_SZ

```
const int MAX_SZ = 20000
```

6.9.1.3 NUM_DIVS

```
const int NUM_DIVS = 32
```

6.10 finder.hh

Go to the documentation of this file.

```
00001 #ifndef FINDER_HH
00002 #define FINDER_HH
00003
00004 #include "geometry.hh"
00005 #include <algorithm>
00006 #include <map>
00007 #include <set>
00008 #include <list>
00009 #include <iostream>
00010
00011
00012 const int NUM_DIVS = 32;
00013 const int MAX_SZ = 20000;
00014 const int DIVIDER = MAX_SZ/NUM_DIVS;
00015
00016
00017
00022 template <typename T>
00023 class Finder {
00024     private:
00025         // Vector de caselles amb els objectes
00026         std::vector< std::set<T *> > _container;
00027
00028         // Set d'apuntadors a les caselles on hi ha l'objecte
00029         std::map<T *, std::set<int> > _locator;
00030
00031         pro2::Pt _divider;
00032         pro2::Rect _range;
00033         pro2::Pt   _num_divs;
00034
00035     public:
00046         Finder (pro2::Rect range = {0,0,MAX_SZ,MAX_SZ}, pro2::Pt divider = {MAX_SZ/NUM_DIVS,
      MAX_SZ/NUM_DIVS}) : _range(range), _divider(divider), _num_divs({(range.right-range.left)/divider.x,
      (range.bottom-range.top)/divider.y})
00047         {
00048             _container = std::vector< std::set<T *> > (_num_divs.x*_num_divs.y);
00049         };
00050
00056         void add(T *t) {
00057             pro2::Rect rect = t->get_rect();
00058             for (int i = (rect.top - _range.top)/_divider.y; i <= (rect.bottom -
      _range.top)/_divider.y; i++) {
00059                 for (int j = (rect.left - _range.left)/_divider.x; j <= (rect.right -
      _range.left)/_divider.x; j++) {
00060                     _container[i*_num_divs.y + j].insert(t);
00061                     _locator[t].insert(i*_num_divs.y + j);
00062                 }
00063             }
00064         };
00065
00073         void update(T *t) {
```

```
00074             remove(t);
00075             add(t);
00076         };
00077
00085         void remove(T *t) {
00086             for (std::set<int>::iterator it = _locator.find(t)->second.begin(); it !=
      _locator.find(t)->second.end(); it++) {
00087                 _container[*it].erase(t);
00088             }
00089             _locator.erase(t);
00090         };
00091
00099         void remove_and_delete(T *t) {
00100             remove(t);
00101             delete t;
00102         }
00103
00109         std::set<T *> query(pro2::Rect rect) const {
00110             std::set<T *> result;
00111             for (int i = (rect.top - _range.top)/_divider.y; i <= (rect.bottom -
      _range.top)/_divider.y; i++) {
00112                 for (int j = (rect.left - _range.left)/_divider.x; j <= (rect.right -
      _range.left)/_divider.x; j++) {
00113                     for (typename std::set<T *>::const_iterator it = _container[i*_num_divs.y +
      j].begin(); it != _container[i*_num_divs.y + j].end(); it++) {
00114                         pro2::Rect obj_rect = (*it)->get_rect();
00115                         if (not(rect.left > obj_rect.right or rect.right < obj_rect.left or rect.top >
      obj_rect.bottom or rect.bottom < obj_rect.top)) result.insert(*it);
00116                     }
00117                 }
00118             }
00119             return result;
00120         };
00121
00125         void AddFromList(std::list<T> &set) {
00126             for (typename std::list<T>::const_iterator it = set.begin(); it != set.end(); it++) {
00127                 add(&(*it));
00128             }
00129         }
00130 };
00131
00132 #endif
```

6.11 game.cc File Reference

```
#include "game.hh"
#include "utils.hh"
```

6.12 game.hh File Reference

```
#include <list>
#include "mario.hh"
#include "platform.hh"
#include "window.hh"
#include "coin.hh"
#include "text.hh"
#include "finder.hh"
#include "block.hh"
#include "interactables.hh"
```

Classes

- class Game

6.13 game.hh

Go to the documentation of this file.

```
00001 #ifndef GAME_HH
00002 #define GAME_HH
00003
00004 #include <list>
00005 #include "mario.hh"
00006 #include "platform.hh"
00007 #include "window.hh"
00008 #include "coin.hh"
00009 #include "text.hh"
00010 #include "finder.hh"
00011 #include "block.hh"
00012 #include "interactables.hh"
00013
00014
00015 class Game {
00016     Mario               mario_;
00017     Finder<Platform>    platforms_;
00018     Finder<Coin>        coins_;
00019     Finder<Block>       blocks_;
00020
00021     std::list<Interactable>    interactables_;
00022
00023     pro2::Rect          death_barrier_;
00024
00025     int curr_anim_frame_ = 0;
00026
00027     bool finished_;
00028     int exit_code_ = 0;
00029     bool paused_;
00030
00031     pro2::TextWriter TW_;
00032
00033     void process_keys(pro2::Window& window);
00034     void update_objects(pro2::Window& window);
00035     void update_camera(pro2::Window& window);
00036
00037  public:
00038     Game(int width, int height, pro2::TextWriter TW, pro2::Rect death_barrier=pro2::Rect{-1000, -2000,
      100000, 400});
00039
00040     void update(pro2::Window& window);
00041     void paint(pro2::Window& window);
00042
00043     bool is_finished() const {
00044         return finished_;
00045     }
00046
00047     bool is_paused() const {
00048         return paused_;
00049     }
00050
00054     int exit_code() const {return exit_code_;}
00055
00059     void spawn_coin(pro2::Pt pos, pro2::DoubPt vel);
00060
00064     void anim_step() {
00065         curr_anim_frame_++;
00066         curr_anim_frame_ %= 18;
00067     }
00068
00069  private:
00070     static constexpr int sky_blue = 0x5c94fc;
00071
00072     static constexpr int anim_len = 10;
00073 };
00074
00075 #endif
```

6.14 geometry.cc File Reference

```
#include "geometry.hh"
```

Namespaces

- namespace pro2

Functions

- std::pair< bool, int > pro2::resolve_collision_vertical (const Rect &prev, Rect curr, const Rect &block)

  Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat des de sobre i -1 altrament.
- std::pair< bool, int > pro2::resolve_collision_horizontal (const Rect &prev, Rect curr, const Rect &block)

  Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat d'esquerra a dreta i -1 altrament.

6.15 geometry.hh File Reference

```
#include <utility>
```

Classes

- struct pro2::Pt
- struct pro2::DoubPt
- struct pro2::Rect

Namespaces

- namespace pro2

Functions

- bool pro2::operator< (const Pt &a, const Pt &b)

  Compara dos punts del pla.
- Pt pro2::round_dpt (const DoubPt &a)

  Retorna un pro2::Pt amb els valors truncats d'un pro2::DoubPt.
- bool pro2::check_collision (const Rect &a, const Rect &b)

  Retorna true si s'intersseccionen els dos pro2::Rect.
- std::pair< bool, int > pro2::resolve_collision_vertical (const Rect &prev, Rect curr, const Rect &block)

  Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat des de sobre i -1 altrament.
- std::pair< bool, int > pro2::resolve_collision_horizontal (const Rect &prev, Rect curr, const Rect &block)

  Retorna true si s'ha interseccionat verticalment i un int amb la direcció: 1 si s'ha creuat d'esquerra a dreta i -1 altrament.

6.16 geometry.hh

Go to the documentation of this file.

```
00001 #ifndef GEOMETRY_HH
00002 #define GEOMETRY_HH
00003
00004 #include <utility>
00005
00006 namespace pro2 {
00007
00008 struct Pt {
00009     int x = 0, y = 0;
00010
00011     Pt operator+(const Pt& other) const {
00012         return {x + other.x, y + other.y};
00013     }
00014     Pt operator-(const Pt& other) const {
00015         return {x - other.x, y - other.y};
00016     }
00017
00018     Pt& operator+=(const Pt& other) {
00019         x += other.x;
00020         y += other.y;
00021         return *this;
00022     }
00023     Pt& operator-=(const Pt& other) {
00024         x -= other.x;
00025         y -= other.y;
00026         return *this;
00027     }
00028 };
00029
00037 inline bool operator<(const Pt& a, const Pt& b) {
00038     return a.x != b.x ? a.x < b.x : a.y < b.y;
00039 }
00040
00041 // Punts amb double
00042 struct DoubPt {
00043     double x = 0, y = 0;
00044 };
00045
00049 inline Pt round_dpt(const DoubPt& a) {
00050     return Pt{int(a.x), int(a.y)};
00051 };
00052
00053 struct Rect {
00054     int left, top, right, bottom;
00055     int width() const {return right - left;}
00056     int height() const {return bottom - top;}
00057
00058     Rect& operator+=(const Rect& other) {
00059         left += other.left;
00060         right += other.right;
00061         top += other.top;
00062         bottom += other.bottom;
00063         return *this;
00064     }
00065
00066     Rect& operator-=(const Rect& other) {
00067         left -= other.left;
00068         right -= other.right;
00069         top -= other.top;
00070         bottom -= other.bottom;
00071         return *this;
00072     }
00073 };
00074
00078 inline bool check_collision(const Rect& a, const Rect& b) {
00079     return ((a.left <= b.left ? a.right >= b.left : a.left <= b.right) and (a.top <= b.top ? a.bottom
      >= b.top : a.top <= b.bottom));
00080 };
00081
00085 std::pair<bool, int> resolve_collision_vertical(const Rect& prev, Rect curr, const Rect& block);
00086
00090 std::pair<bool, int> resolve_collision_horizontal(const Rect& prev, Rect curr, const Rect& block);
00091
00092 }
00093
00094 #endif
```

6.17 interactables.cc File Reference

```
#include "interactables.hh"
#include "utils.hh"
```

6.18 interactables.hh File Reference

```
#include <vector>
#include "geometry.hh"
#include "finder.hh"
#include "platform.hh"
#include "block.hh"
#include "utils.hh"
#include "window.hh"
```

Classes

- class Interactable

  La classe Interactable engloba tot tipus d'objectes que tenen un comportament diferent amb els que el mario pot interactuar.

6.19 interactables.hh

Go to the documentation of this file.

```
00001 #ifndef INTERACTABLES_HH
00002 #define INTERACTABLES_HH
00003
00004 #include <vector>
00005 #include "geometry.hh"
00006 #include "finder.hh"
00007 #include "platform.hh"
00008 #include "block.hh"
00009 #include "utils.hh"
00010 #include "window.hh"
00011
00029 class Interactable {
00030     private:
00031         int type_; // 0 per moneda, 1 per bolet, 2 per bandera, 3 per goomba
00032
00033         pro2::Pt pos_;
00034         int direction = 1;
00035
00036         int despawn_timer_;
00037
00038     public:
00039         Interactable(pro2::Pt pos, int type) : pos_(pos), type_(type), despawn_timer_((type==0)? 10 :
      -1) {};
00040
00041         bool update(pro2::Window& window, const Finder<Platform>& platforms, const Finder<Block>&
      blocks);
00042
00043         void paint(pro2::Window& window) const;
00044
00045         pro2::Rect collision_box() const {
00046             if (type_ == 0) return {0,0,0,0};
00047             else if (type_ == 1 or type_ == 3) return {pos_.x, pos_.y, pos_.x + 15, pos_.y + 15};
00048             else return {pos_.x+7, pos_.y, pos_.x + 8, pos_.y + 168};
00049         }
00050
00051         int type() const {return type_;}
00052
```

```
00053         pro2::Pt pos() const {return pos_;}
00054
00055     private:
00056         static const std::vector<std::vector<std::vector<int>> sprites;
00057 };
00058
00059 #endif
```

6.20 main.cc File Reference

```
#include <vector>
#include "game.hh"
#include "window.hh"
#include "start_screen.hh"
```

Functions

- void death_screen (pro2::Window &window, pro2::TextWriter &tw)

  Dibuixa la pantalla de mort i espera 5 segons (o fins que es prem 'Escape')
- void win_screen (pro2::Window &window, pro2::TextWriter &tw)

  Dibuixa la pantalla de victòria i espera 5 segons (o fins que es prem 'Escape')
- int main ()

Variables

- const int WIDTH = 480
- const int HEIGHT = 320
- const int ZOOM = 2
- const int FPS = 48

6.20.1 Function Documentation

6.20.1.1 death_screen()

```
void death_screen (
            pro2::Window & window,
            pro2::TextWriter & tw)
```

Dibuixa la pantalla de mort i espera 5 segons (o fins que es prem 'Escape')

6.20.1.2 main()

```
int main ()
```

6.20.1.3 win_screen()

```
void win_screen (
             pro2::Window & window,
             pro2::TextWriter & tw)
```

Dibuixa la pantalla de victòria i espera 5 segons (o fins que es prem 'Escape')

6.20.2 Variable Documentation

6.20.2.1 FPS

```
const int FPS = 48
```

6.20.2.2 HEIGHT

```
const int HEIGHT = 320
```

6.20.2.3 WIDTH

```
const int WIDTH = 480
```

6.20.2.4 ZOOM

```
const int ZOOM = 2
```

6.21 mario.cc File Reference

```
#include "mario.hh"
#include "utils.hh"
```

6.22 mario.hh File Reference

```
#include <iostream>
#include <list>
#include "platform.hh"
#include "window.hh"
#include "block.hh"
#include "finder.hh"
#include "interactables.hh"
```

Classes

- class Mario

6.23 mario.hh

Go to the documentation of this file.

```
00001 #ifndef MARIO_HH
00002 #define MARIO_HH
00003
00004 #include <iostream>
00005 #include <list>
00006 #include "platform.hh"
00007 #include "window.hh"
00008 #include "block.hh"
00009 #include "finder.hh"
00010 #include "interactables.hh"
00011
00012 class Mario {
00013  private:
00014     pro2::Pt pos_, last_pos_;
00015     pro2::Pt speed_ = {0, 0};
00016     pro2::Pt accel_ = {0, 0};
00017     int      accel_time_ = 0;
00018
00019     int key_up_;
00020     int key_left_;
00021     int key_right_;
00022
00023     bool grounded_ = true;
00024     bool looking_left_ = false;
00025
00026     int coin_count_ = 0;
00027
00028     int state_ = 0;
00029     int anim_frame_counter_;
00030
00031  public:
00032     Mario(int key_up, int key_left, int key_right, pro2::Pt pos):
00033         key_up_(key_up), key_left_(key_left), key_right_(key_right),
00034         pos_(pos), last_pos_(pos)
00035         {}
00036
00037     void paint(pro2::Window& window) const;
00038
00039     pro2::Rect collision_box() const;
00040
00041     pro2::Pt pos() const {
00042         return pos_;
00043     }
00044
00045     void set_y(int y) {
00046         pos_.y = y;
00047     }
00048
00049     bool is_grounded() const {
00050         return grounded_;
00051     }
00052
00053     void set_grounded(bool grounded) {
00054         grounded_ = grounded;
00055         if (grounded_) {
00056             speed_.y = 0;
00057         }
00058     }
00059
00060     void toggle_grounded() {
00061         set_grounded(!grounded_);
00062     }
00063
00064     void jump();
00065
00071     void add_coin(int ammount = 1) {
00072         coin_count_ += ammount;
00073     }
00074
00078     int get_coin_count() {
00079         return coin_count_;
00080     }
00081
```

```
00082     void update(pro2::Window& window, const Finder<Platform>& platforms, Finder<Block>& blocks,
      std::list<Interactable>& interactables);
00083
00090     void set_state(int new_state);
00091
00095     int get_state() const {return state_;}
00096
00097  private:
00098     static const std::vector<std::vector<int>> mario_sprite_normal_;
00099     static const std::vector<std::vector<int>> mario_sprite_tall_;
00100 };
00101
00102 #endif
```

6.24 platform.cc File Reference

```
#include "platform.hh"
```

Typedefs

- typedef uint32_t Color

6.24.1 Typedef Documentation

6.24.1.1 Color

```
typedef uint32_t pro2::Color
```

6.25 platform.hh File Reference

```
#include <vector>
#include "window.hh"
```

Classes

- class Platform

6.26 platform.hh

Go to the documentation of this file.

```
00001 #ifndef PLATFORM_HH
00002 #define PLATFORM_HH
00003
00004 #include <vector>
00005 #include "window.hh"
00006
00007 class Platform {
00008  private:
00009     int left_, right_, top_, bottom_;
00010
00011     static const std::vector<std::vector<int» platform_texture_;
00012
00013  public:
00014     Platform() : left_(0), right_(0), top_(0), bottom_(0) {}
00015
00016     Platform(const Platform& other)
00017         : left_(other.left_), right_(other.right_), top_(other.top_), bottom_(other.bottom_) {}
00018
00019     Platform(int left, int right, int top, int bottom)
00020         : left_(left), right_(right), top_(top), bottom_(bottom) {}
00021
00022     void paint(pro2::Window& window) const;
00023     bool has_crossed_floor_downwards(pro2::Pt plast, pro2::Pt pcurr) const;
00024     bool is_pt_inside(pro2::Pt pt) const;
00025
00026     int top() const {
00027         return top_;
00028     }
00029
00030     pro2::Rect get_rect() const {
00031         return {left_, top_, right_, bottom_};
00032     }
00033 };
00034
00035 #endif
```

6.27 start_screen.cc File Reference

```
#include "start_screen.hh"
```

6.28 start_screen.hh File Reference

```
#include <vector>
#include <string>
#include "geometry.hh"
#include "window.hh"
#include "text.hh"
#include "utils.hh"
```

Classes

- class Button

  Classe que implementa botons i la seva funcionalitat.
- class StartScreen

  Aquesta classe s'encarrega de dibuixar el menú principal i gestionar la interacció de l'usuari.

6.29 start_screen.hh

Go to the documentation of this file.

```
00001 #ifndef START_SCREEN_HH
00002 #define START_SCREEN_HH
00003
00004 #include <vector>
00005 #include <string>
00006 #include "geometry.hh"
00007 #include "window.hh"
00008 #include "text.hh"
00009 #include "utils.hh"
00010
00011
00017 class Button {
00018     private:
00019         pro2::Rect rect;
00020         std::string text;
00021         int bg_normal, bg_selected;
00022     public:
00023         Button(
00024             pro2::Rect rect,
00025             std::string text,
00026             int bg_normal=0xCD612E,
00027             int bg_selected=0xC97A55
00028         ) : rect(rect), text(text), bg_normal(bg_normal), bg_selected(bg_selected) {};
00029
00030         pro2::Rect get_rect() const {return rect;}
00031
00035         bool selected(pro2::Pt pos) const {return (pos.x >= rect.left and pos.x <= rect.right and
      pos.y >= rect.top and pos.y <= rect.bottom);}
00036
00037         void paint(pro2::Window& window, pro2::TextWriter& writer) const;
00038 };
00039
00040
00047 class StartScreen {
00048     private:
00049         std::vector< Button > buttons;
00050         pro2::TextWriter TW_;
00051
00052         // Flags
00053         bool finished_ = false;
00054         int exit_code_ = 0;
00055
00056         int height_;
00057         int width_;
00058
00059     public:
00060         StartScreen(int width, int height, pro2::TextWriter TW);
00061
00062         inline bool is_finished() const {return finished_;}
00063
00064         void process_keys(pro2::Window& window);
00065
00066         void update(pro2::Window& window);
00067
00068         void paint(pro2::Window& window);
00069
00073         int exit_code() const {return exit_code_;}
00074
00078         void restart(pro2::Window& window);
00079
00080     private:
00081         static constexpr int sky_blue = 0x5c94fc;
00082 };
00083
00084
00085 #endif
```

6.30 text.cc File Reference

```
#include "text.hh"
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
```

```
#include <sstream>
#include <map>
```

Namespaces

- namespace pro2

Functions

- std::ifstream pro2::read_file (std::string fname)

  Carrega un fitxer de text com a 'stream'.
- Palette pro2::read_colors (std::ifstream &stream)

  Genera una paleta de colors a partir d'un fitxer de text.
- Font pro2::read_sprites (std::ifstream &stream)

  Genera una font (vector de sprites de tots els caràcters) a partir d'un fitxer de text Els fitxers de tipus font tenen la següent estructura: A la primera línia hi ha 3 'int' amb 'count' (nombre de sprites), 'height' i 'width' (dels sprites) A continuació hi ha tots els sprites amb els caràcters.
- Charset pro2::read_charset (std::ifstream &stream)

  Carrega el charset (els caràcters disponibles i l'ordre corresponent) de la font a partir d'un fitxer de text.
- Sprite pro2::color_sprite (const std::vector< std::vector< std::string > > &sprite, const Palette &colors)

  Pinta el caràcter de la font amb una paleta de colors.
- void pro2::paint_char (Window &window, Pt &pos, Sprite sprite, int &size)

  Pinta a la pantalla un caràcter amb una posició i tamany.
- std::vector< std::string > pro2::split_lines (std::string text)

  *Separa un string multilínia (separades per '
  ') en un vector de strings per cada línia.*

6.31 text.hh File Reference

```
#include <iostream>
#include <fstream>
#include <map>
#include "utils.hh"
#include "window.hh"
#include "geometry.hh"
```

Classes

- class pro2::TextWriter

Namespaces

- namespace pro2

Typedefs

- typedef std::vector< std::vector< std::vector< std::string > > > pro2::Font

  Vector de caràcters. Els caràcters són matrius de strings. Cada element de la matriu representa un pixel a pintar, i cada string diferent està mapejada a la paleta amb el color que li correspon.
- typedef std::map< std::string, int > pro2::Palette

  Mapa de strings a colors.
- typedef std::map< char, int > pro2::Charset

  Mapa de caràcters a l'índex de l'sprite de la font que li correspon.

Functions

- std::ifstream pro2::read_file (std::string fname)

  Carrega un fitxer de text com a 'stream'.
- Palette pro2::read_colors (std::ifstream &stream)

  Genera una paleta de colors a partir d'un fitxer de text.
- Font pro2::read_sprites (std::ifstream &stream)

  Genera una font (vector de sprites de tots els caràcters) a partir d'un fitxer de text Els fitxers de tipus font tenen la següent estructura: A la primera línia hi ha 3 'int' amb 'count' (nombre de sprites), 'height' i 'width' (dels sprites) A continuació hi ha tots els sprites amb els caràcters.
- Charset pro2::read_charset (std::ifstream &stream)

  Carrega el charset (els caràcters disponibles i l'ordre corresponent) de la font a partir d'un fitxer de text.
- Sprite pro2::color_sprite (const std::vector< std::vector< std::string > > &sprite, const Palette &colors)

  Pinta el caràcter de la font amb una paleta de colors.
- void pro2::paint_char (Window &window, Pt &pos, Sprite sprite, int &size)

  Pinta a la pantalla un caràcter amb una posició i tamany.
- std::vector< std::string > pro2::split_lines (std::string text)

  *Separa un string multilínia (separades per '
  ') en un vector de strings per cada línia.*

6.32 text.hh

Go to the documentation of this file.

```
00001 #ifndef TEXT_HH
00002 #define TEXT_HH
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <map>
00007 #include "utils.hh"
00008 #include "window.hh"
00009 #include "geometry.hh"
00010
00011
00012 namespace pro2 {
00018     typedef std::vector<std::vector<std::vector<std::string>> Font;
00019
00023     typedef std::map<std::string, int> Palette;
00027     typedef std::map<char, int> Charset;
00028
00034     std::ifstream read_file(std::string fname);
00035
00041     Palette read_colors(std::ifstream &stream);
00042
00051     Font read_sprites(std::ifstream &stream);
00052
00058     Charset read_charset(std::ifstream &stream);
00059
00066     Sprite color_sprite(const std::vector<std::vector<std::string>> &sprite, const Palette &colors);
00067
00068
```

```
00077     void paint_char(Window& window, Pt& pos, Sprite sprite, int& size);
00078
00084     std::vector<std::string> split_lines(std::string text);
00085
00091     class TextWriter {
00092         private:
00093             Font font_;
00094             Palette palette_;
00095             Charset charset_;
00096
00097         public:
00098             TextWriter(Font font, Palette palette) : font_(font), palette_(palette) {};
00099             TextWriter(Font font, std::string palette_path);
00100             TextWriter(std::string font_path, Palette palette);
00101             TextWriter(std::string font_path, std::string palette_path);
00102
00109             Sprite get_sprite(char ch) const;
00110
00111             void set_font(Font font) {font_ = font;}
00112             void set_font(std::string path);
00113             const Font get_font() {return font_;};
00114
00115             void set_palette(Palette palette) {palette_ = palette;}
00116             void set_palette(std::string path);
00117             const Palette get_palette() {return palette_;};
00118
00119             void set_charset(Charset charset) {charset_ = charset;}
00120             void set_charset(std::string path);
00121             const Charset get_charset() {return charset_;};
00122
00135             void write_text(Window& window, const Pt& orig, const std::string& text, int
      space_between_chars=1, int size=4, Pt alignment={0,0});
00136         };
00137     }
00138 #endif
```

6.33 utils.cc File Reference

```
#include "utils.hh"
#include <iostream>
```

Namespaces

- namespace pro2

Functions

- double pro2::random_double (int min, int max, int precision)

  Retorna un double entre (min, max), amb una precisó de n digits.
- void pro2::paint_hline (Window &window, int xini, int xfin, int y, Color color=white)

  Dibuja una línea horizontal en la ventana.
- void pro2::paint_vline (Window &window, int x, int yini, int yfin, Color color=white)

  Dibuja una línea vertical en la ventana.
- void pro2::paint_rect (Window &window, Rect rect, Color color, int brush_sz)

  Dibuixa un rectangle 'Rect'.
- void pro2::paint_rect_fill (Window &window, Rect rect, Color color)

  Dibuixa i emplena un rectangle 'Rect'.
- void pro2::paint_rect_fill_transparent (Window &window, Rect rect, Color color, double transp)

  Dibuixa i emplena un rectangle 'Rect' amb transparència.
- void pro2::paint_pixel_transparent (Window &window, Pt pos, Color color, double transp)

  Pinta un pixel amb color i transparència.
- void pro2::paint_sprite (Window &window, Pt orig, const Sprite &sprite, bool mirror)

  Dibuixa una imatge/textura a la finestra a partir d'una posició

6.34 utils.hh File Reference

```
#include <vector>
#include <cstdlib>
#include "geometry.hh"
#include "window.hh"
```

Namespaces

- namespace pro2

Typedefs

- typedef std::vector< std::vector< int > > pro2::Sprite

Functions

- double pro2::random_double (int min, int max, int precision)

  Retorna un double entre (min, max), amb una precisó de n digits.
- void pro2::paint_hline (Window &window, int xini, int xfin, int y, Color color=white)

  Dibuja una línea horizontal en la ventana.
- void pro2::paint_vline (Window &window, int x, int yini, int yfin, Color color=white)

  Dibuja una línea vertical en la ventana.
- void pro2::paint_rect (Window &window, Rect rect, Color color, int brush_sz)

  Dibuixa un rectangle 'Rect'.
- void pro2::paint_rect_fill (Window &window, Rect rect, Color color)

  Dibuixa i emplena un rectangle 'Rect'.
- void pro2::paint_rect_fill_transparent (Window &window, Rect rect, Color color, double transp)

  Dibuixa i emplena un rectangle 'Rect' amb transparència.
- void pro2::paint_pixel_transparent (Window &window, Pt pos, Color color, double transp)

  Pinta un pixel amb color i transparència.
- void pro2::paint_sprite (Window &window, Pt orig, const Sprite &sprite, bool mirror)

  Dibuixa una imatge/textura a la finestra a partir d'una posició

6.35 utils.hh

Go to the documentation of this file.

```
00001 #ifndef UTILS_HH
00002 #define UTILS_HH
00003
00004 #include <vector>
00005 #include <cstdlib>          // Per als valors aleatoris
00006 #include "geometry.hh"
00007 #include "window.hh"
00008
00009 namespace pro2 {
00010
00011 typedef std::vector<std::vector<int>> Sprite;
00012
00020 double random_double(int min, int max, int precision);
00021
00031 void paint_hline(Window& window, int xini, int xfin, int y, Color color = white);
00032
```

```
00042 void paint_vline(Window& window, int x, int yini, int yfin, Color color = white);
00043
00052 void paint_rect(
00053     Window& window,
00054     Rect    rect,
00055     Color   color,
00056     int     brush_sz
00057 );
00058
00066 void paint_rect_fill(
00067     Window& window,
00068     Rect    rect,
00069     Color   color
00070 );
00071
00080 void paint_rect_fill_transparent(
00081     Window& window,
00082     Rect    rect,
00083     Color   color,
00084     double  transp
00085 );
00086
00096 void paint_pixel_transparent(
00097     Window& window,
00098     Pt      pos,
00099     Color   color,
00100     double  transp
00101 );
00102
00111 void paint_sprite(Window&           window,
00112                   Pt                orig,
00113                   const Sprite&     sprite,
00114                   bool              mirror);
00115 }
00116
00117 // namespace pro2
00118
00119 #endif
```

6.36 window.cc File Reference

```
#include "fenster.h"
#include "window.hh"
```

Namespaces

- namespace pro2

6.37 window.hh File Reference

```
#include <cassert>
#include <string>
#include "fenster.h"
#include "geometry.hh"
```

Classes

- class pro2::Window

Namespaces

- namespace pro2

Macros

- #define FENSTER_HEADER

Typedefs

- typedef uint32_t pro2::Color

Enumerations

- enum pro2::ModKey { pro2::Ctrl = 1 , pro2::Shift = 2 , pro2::Alt = 4 , pro2::Meta = 8 }
- enum pro2::Keys {
  pro2::Space = 32 , pro2::Backspace = 8 , pro2::Delete = 127 , pro2::End = 5 ,
  pro2::Escape = 27 , pro2::Home = 2 , pro2::Insert = 26 , pro2::PageDown = 4 ,
  pro2::PageUp = 3 , pro2::Return = 10 , pro2::Tab = 9 , pro2::Up = 17 ,
  pro2::Down = 18 , pro2::Right = 19 , pro2::Left = 20 }

Variables

- const Color pro2::black = 0x00000000
- const Color pro2::red = 0x00ff0000
- const Color pro2::green = 0x0000ff00
- const Color pro2::blue = 0x000000ff
- const Color pro2::yellow = 0x00ffff00
- const Color pro2::magenta = 0x00ff00ff
- const Color pro2::cyan = 0x0000ffff
- const Color pro2::white = 0x00ffffff

6.37.1 Macro Definition Documentation

6.37.1.1 FENSTER_HEADER

```
#define FENSTER_HEADER
```

6.38 window.hh

Go to the documentation of this file.

```
00001 #ifndef WINDOW_HH
00002 #define WINDOW_HH
00003
00004 #include <cassert>
00005 #include <string>
00006
00007 #define FENSTER_HEADER
00008 #include "fenster.h"
00009
00010 #include "geometry.hh"
00011
00012 namespace pro2 {
00013
00019 enum ModKey { Ctrl = 1, Shift = 2, Alt = 4, Meta = 8 };
00020
00026
00027 typedef uint32_t Color;
00028
00029 const Color black = 0x00000000;
00030 const Color red = 0x00ff0000;
00031 const Color green = 0x0000ff00;
00032 const Color blue = 0x000000ff;
00033 const Color yellow = 0x00ffff00;
00034 const Color magenta = 0x00ff00ff;
00035 const Color cyan = 0x0000ffff;
00036 const Color white = 0x00ffffff;
00037
00044 enum Keys {
00045     Space = 32,
00046     Backspace = 8,
00047     Delete = 127,
00048     End = 5,
00049     Escape = 27,
00050     Home = 2,
00051     Insert = 26,
00052     PageDown = 4,
00053     PageUp = 3,
00054     Return = 10,
00055     Tab = 9,
00056     // Arrows,
00057     Up = 17,
00058     Down = 18,
00059     Right = 19,
00060     Left = 20,
00061 };
00062
00070 class Window {
00071  private:
00075     int     last_keys_[256];
00076     int     last_mouse_;
00077     fenster fenster_;
00078
00085     uint32_t *pixels_;
00086
00090     size_t pixels_size_;
00091
00095     int zoom_ = 1;
00096
00100     int64_t last_time_;
00101
00105     int frame_count_ = 0;
00106
00110     uint8_t fps_ = 60;
00111
00112     // Cámara
00113
00117     Pt topleft_ = {0, 0};
00118     Pt topleft_target_ = {0, 0};
00119
00123     void update_camera_();
00124
00129     bool camera_moving_() const {
00130         return topleft_.x != topleft_target_.x || topleft_.y != topleft_target_.y;
00131     }
00132
00136     static constexpr int camera_speed_ = 8;
00137
00138  public:
00154     Window(std::string title, int width, int height, int zoom = 1);
00155
00160     ~Window() {
00161         fenster_close(&fenster_);
```

```
00162         delete[] pixels_;
00163     }
00164
00169     int width() const {
00170         return fenster_.width / zoom_;
00171     }
00172
00177     int height() const {
00178         return fenster_.height / zoom_;
00179     }
00180
00216     bool next_frame();
00217
00230     void clear(Color color = black);
00231
00245     int frame_count() const {
00246         return frame_count_;
00247     }
00248
00273     bool is_key_down(int code) const {
00274         return code >= 0 && code < 128 && fenster_.keys[code];
00275     }
00276
00297     bool was_key_pressed(int code) const {
00298         return code >= 0 && code < 128 && !last_keys_[code] && fenster_.keys[code];
00299     }
00300
00314     bool is_modkey_down(ModKey key) const {
00315         return fenster_.mod & uint8_t(key);
00316     }
00317
00327     bool is_mouse_down() const {
00328         return bool(fenster_.mouse);
00329     }
00330
00340     bool was_mouse_pressed() const {
00341         return !last_mouse_ && bool(fenster_.mouse);
00342     }
00343
00350     Pt mouse_pos() const;
00351
00360     void sleep(int ms) const {
00361         fenster_sleep(ms);
00362     }
00363
00370     Color get_pixel(Pt xy) const {
00371         return fenster_pixel(&fenster_, xy.x * zoom_, xy.y * zoom_);
00372     }
00373
00386     void set_pixel(Pt xy, Color color);
00387
00400     void set_fps(int fps) {
00401         assert(fps > 0 && fps < 240);
00402         fps_ = fps;
00403     }
00404
00414     void move_camera(Pt desplazamiento) {
00415         if (!camera_moving_()) {
00416             topleft_target_.x = topleft_.x + desplazamiento.x;
00417             topleft_target_.y = topleft_.y + desplazamiento.y;
00418         }
00419     }
00420
00426     Pt camera_center() const {
00427         const int width = fenster_.width / zoom_;
00428         const int height = fenster_.height / zoom_;
00429         return {topleft_.x + width / 2, topleft_.y + height / 2};
00430     }
00431
00432     Rect camera_rect() const {
00433         const int width = fenster_.width / zoom_;
00434         const int height = fenster_.height / zoom_;
00435         const int left = topleft_.x;
00436         const int top = topleft_.y;
00437         const int right = topleft_.x + width;
00438         const int bottom = topleft_.y + height;
00439         return {left, top, right, bottom};
00440     }
00441
00449     void set_camera_topleft(Pt topleft) {
00450         topleft_ = topleft;
00451         topleft_target_ = topleft;
00452     }
00453
00459     Pt topleft() const {
00460         return topleft_;
00461     }
```

```
00462 };
00463
00464 }  // namespace pro2
00465
00466 #endif
```

Index