

Super Mario Pro2

Generated by Doxygen 1.14.0

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 pro2 Namespace Reference	7
4.1.1 Typedef Documentation	9
4.1.1.1 Charset	9
4.1.1.2 Color	9
4.1.1.3 Font	9
4.1.1.4 Palette	9
4.1.1.5 Sprite	9
4.1.2 Enumeration Type Documentation	9
4.1.2.1 Keys	9
4.1.2.2 ModKey	10
4.1.3 Function Documentation	10
4.1.3.1 check_collision()	10
4.1.3.2 color_sprite()	10
4.1.3.3 operator<()	10
4.1.3.4 paint_char()	10
4.1.3.5 paint_hline()	11
4.1.3.6 paint_pixel_transparent()	11
4.1.3.7 paint_rect()	11
4.1.3.8 paint_rect_fill()	12
4.1.3.9 paint_rect_fill_transparent()	12
4.1.3.10 paint_sprite()	12
4.1.3.11 paint_vline()	13
4.1.3.12 random_double()	13
4.1.3.13 read_charset()	13
4.1.3.14 read_colors()	14
4.1.3.15 read_file()	14
4.1.3.16 read_sprites()	14
4.1.3.17 resolve_collision_horizontal()	15
4.1.3.18 resolve_collision_vertical()	15
4.1.3.19 round_dpt()	15
4.1.3.20 split_lines()	15
4.1.4 Variable Documentation	16
4.1.4.1 black	16

4.1.4.2 blue	16
4.1.4.3 cyan	16
4.1.4.4 green	16
4.1.4.5 magenta	16
4.1.4.6 red	16
4.1.4.7 white	16
4.1.4.8 yellow	16
5 Class Documentation	17
5.1 Block Class Reference	17
5.1.1 Constructor & Destructor Documentation	17
5.1.1.1 Block() [1/2]	17
5.1.1.2 Block() [2/2]	18
5.1.2 Member Function Documentation	18
5.1.2.1 block_type()	18
5.1.2.2 check_bumped()	18
5.1.2.3 get_rect()	18
5.1.2.4 get_sprite()	18
5.1.2.5 paint()	19
5.1.2.6 pos()	19
5.1.3 Member Data Documentation	19
5.1.3.1 sprites	19
5.2 Button Class Reference	19
5.2.1 Detailed Description	19
5.2.2 Constructor & Destructor Documentation	20
5.2.2.1 Button()	20
5.2.3 Member Function Documentation	20
5.2.3.1 get_rect()	20
5.2.3.2 paint()	20
5.2.3.3 selected()	20
5.3 Coin Class Reference	20
5.3.1 Constructor & Destructor Documentation	21
5.3.1.1 Coin()	21
5.3.2 Member Function Documentation	21
5.3.2.1 get_rect()	21
5.3.2.2 get_sprite()	22
5.3.2.3 is_grounded()	22
5.3.2.4 paint()	22
5.3.2.5 pos()	22
5.3.2.6 set_grounded()	22
5.3.2.7 set_y()	22
5.3.2.8 toggle_grounded()	22

5.3.2.9 update()	22
5.3.3 Member Data Documentation	23
5.3.3.1 sprites	23
5.4 pro2::DoubPt Struct Reference	23
5.4.1 Member Data Documentation	23
5.4.1.1 x	23
5.4.1.2 y	23
5.5 fenster Struct Reference	23
5.5.1 Member Data Documentation	24
5.5.1.1 buf	24
5.5.1.2 dpy	24
5.5.1.3 gc	24
5.5.1.4 height	24
5.5.1.5 img	24
5.5.1.6 keys	24
5.5.1.7 mod	24
5.5.1.8 mouse	25
5.5.1.9 title	25
5.5.1.10 w	25
5.5.1.11 width	25
5.5.1.12 x	25
5.5.1.13 y	25
5.6 Finder< T > Class Template Reference	25
5.6.1 Detailed Description	26
5.6.2 Constructor & Destructor Documentation	26
5.6.2.1 Finder()	26
5.6.3 Member Function Documentation	26
5.6.3.1 add()	26
5.6.3.2 AddFromList()	27
5.6.3.3 query()	27
5.6.3.4 remove()	27
5.6.3.5 remove_and_delete()	27
5.6.3.6 update()	28
5.7 Game Class Reference	29
5.7.1 Constructor & Destructor Documentation	29
5.7.1.1 Game()	29
5.7.2 Member Function Documentation	29
5.7.2.1 anim_step()	29
5.7.2.2 exit_code()	30
5.7.2.3 is_finished()	30
5.7.2.4 is_paused()	30
5.7.2.5 paint()	30

5.7.2.6 spawn_coin()	30
5.7.2.7 update()	30
5.8 Interactable Class Reference	30
5.8.1 Detailed Description	31
5.8.2 Constructor & Destructor Documentation	31
5.8.2.1 Interactable()	31
5.8.3 Member Function Documentation	31
5.8.3.1 collision_box()	31
5.8.3.2 paint()	31
5.8.3.3 pos()	31
5.8.3.4 type()	32
5.8.3.5 update()	32
5.9 Mario Class Reference	32
5.9.1 Constructor & Destructor Documentation	33
5.9.1.1 Mario()	33
5.9.2 Member Function Documentation	33
5.9.2.1 add_coin()	33
5.9.2.2 collision_box()	33
5.9.2.3 get_coin_count()	33
5.9.2.4 get_state()	33
5.9.2.5 is_grounded()	33
5.9.2.6 jump()	33
5.9.2.7 paint()	34
5.9.2.8 pos()	34
5.9.2.9 set_grounded()	34
5.9.2.10 set_state()	34
5.9.2.11 set_y()	34
5.9.2.12 toggle_grounded()	34
5.9.2.13 update()	34
5.10 Platform Class Reference	35
5.10.1 Constructor & Destructor Documentation	35
5.10.1.1 Platform() [1/3]	35
5.10.1.2 Platform() [2/3]	35
5.10.1.3 Platform() [3/3]	35
5.10.2 Member Function Documentation	35
5.10.2.1 get_rect()	35
5.10.2.2 has_crossed_floor_downwards()	35
5.10.2.3 is_pt_inside()	36
5.10.2.4 paint()	36
5.10.2.5 top()	36
5.11 pro2::Pt Struct Reference	36
5.11.1 Member Function Documentation	36

5.11.1.1 operator+()	36
5.11.1.2 operator+=()	37
5.11.1.3 operator-()	37
5.11.1.4 operator-=()	37
5.11.2 Member Data Documentation	37
5.11.2.1 x	37
5.11.2.2 y	37
5.12 pro2::Rect Struct Reference	37
5.12.1 Member Function Documentation	38
5.12.1.1 height()	38
5.12.1.2 operator+=()	38
5.12.1.3 operator-=()	38
5.12.1.4 width()	38
5.12.2 Member Data Documentation	38
5.12.2.1 bottom	38
5.12.2.2 left	38
5.12.2.3 right	38
5.12.2.4 top	38
5.13 StartScreen Class Reference	39
5.13.1 Detailed Description	39
5.13.2 Constructor & Destructor Documentation	39
5.13.2.1 StartScreen()	39
5.13.3 Member Function Documentation	39
5.13.3.1 exit_code()	39
5.13.3.2 is_finished()	39
5.13.3.3 paint()	40
5.13.3.4 process_keys()	40
5.13.3.5 restart()	40
5.13.3.6 update()	40
5.14 pro2::TextWriter Class Reference	40
5.14.1 Detailed Description	41
5.14.2 Constructor & Destructor Documentation	41
5.14.2.1 TextWriter() [1/4]	41
5.14.2.2 TextWriter() [2/4]	41
5.14.2.3 TextWriter() [3/4]	41
5.14.2.4 TextWriter() [4/4]	41
5.14.3 Member Function Documentation	41
5.14.3.1 get_charset()	41
5.14.3.2 get_font()	41
5.14.3.3 get_palette()	41
5.14.3.4 get_sprite()	41
5.14.3.5 set_charset() [1/2]	42

5.14.3.6 set_charset() [2/2]	42
5.14.3.7 set_font() [1/2]	42
5.14.3.8 set_font() [2/2]	42
5.14.3.9 set_palette() [1/2]	42
5.14.3.10 set_palette() [2/2]	42
5.14.3.11 write_text()	42
5.15 pro2::Window Class Reference	43
5.15.1 Detailed Description	44
5.15.2 Constructor & Destructor Documentation	44
5.15.2.1 Window()	44
5.15.2.2 ~Window()	45
5.15.3 Member Function Documentation	45
5.15.3.1 camera_center()	45
5.15.3.2 camera_rect()	45
5.15.3.3 clear()	45
5.15.3.4 frame_count()	45
5.15.3.5 get_pixel()	45
5.15.3.6 height()	46
5.15.3.7 is_key_down()	46
5.15.3.8 is_modkey_down()	46
5.15.3.9 is_mouse_down()	47
5.15.3.10 mouse_pos()	47
5.15.3.11 move_camera()	47
5.15.3.12 next_frame()	48
5.15.3.13 set_camera_topleft()	48
5.15.3.14 set_fps()	48
5.15.3.15 set_pixel()	49
5.15.3.16 sleep()	49
5.15.3.17 topleft()	49
5.15.3.18 was_key_pressed()	50
5.15.3.19 was_mouse_pressed()	50
5.15.3.20 width()	50
6 File Documentation	51
6.1 block.cc File Reference	51
6.2 block.hh File Reference	51
6.3 block.hh	51
6.4 coin.cc File Reference	52
6.5 coin.hh File Reference	53
6.6 coin.hh	53
6.7 fenster.h File Reference	54
6.7.1 Macro Definition Documentation	54

6.7.1.1 _DEFAULT_SOURCE	54
6.7.1.2 FENSTER_API	55
6.7.1.3 fenster_pixel	55
6.7.2 Function Documentation	55
6.7.2.1 fenster_close()	55
6.7.2.2 fenster_loop()	55
6.7.2.3 fenster_open()	55
6.7.2.4 fenster_sleep()	55
6.7.2.5 fenster_time()	55
6.8 fenster.h	56
6.9 finder.hh File Reference	60
6.9.1 Variable Documentation	61
6.9.1.1 DIVIDER	61
6.9.1.2 MAX_SZ	61
6.9.1.3 NUM_DIVS	61
6.10 finder.hh	61
6.11 game.cc File Reference	62
6.12 game.hh File Reference	62
6.13 game.hh	63
6.14 geometry.cc File Reference	63
6.15 geometry.hh File Reference	64
6.16 geometry.hh	65
6.17 interactables.cc File Reference	66
6.18 interactables.hh File Reference	66
6.19 interactables.hh	66
6.20 main.cc File Reference	67
6.20.1 Function Documentation	67
6.20.1.1 death_screen()	67
6.20.1.2 main()	67
6.20.1.3 win_screen()	67
6.20.2 Variable Documentation	68
6.20.2.1 FPS	68
6.20.2.2 HEIGHT	68
6.20.2.3 WIDTH	68
6.20.2.4 ZOOM	68
6.21 mario.cc File Reference	68
6.22 mario.hh File Reference	68
6.23 mario.hh	69
6.24 platform.cc File Reference	70
6.24.1 Typedef Documentation	70
6.24.1.1 Color	70
6.25 platform.hh File Reference	70

6.26 platform.hh	70
6.27 start_screen.cc File Reference	71
6.28 start_screen.hh File Reference	71
6.29 start_screen.hh	71
6.30 text.cc File Reference	72
6.31 text.hh File Reference	73
6.32 text.hh	74
6.33 utils.cc File Reference	75
6.34 utils.hh File Reference	75
6.35 utils.hh	76
6.36 window.cc File Reference	77
6.37 window.hh File Reference	77
6.37.1 Macro Definition Documentation	78
6.37.1.1 FENSTER_HEADER	78
6.38 window.hh	78
Index	81

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

pro2	7
--------------------------------	-------------------

Chapter 2

Class Index

2.1 Class List

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

Block	17
Button	
Class which implements buttons and their functionality The buttons can be clicked, and when the cursor goes over them the background color can be changed (bg_normal/bg_selected)	19
Coin	20
pro2::DoubPt	23
fenster	23
Finder< T >	25
Game	29
Interactable	
The classs Interactable is used for every type of object which interacti with Mario	30
Mario	32
Platform	35
pro2::Pt	36
pro2::Rect	37
StartScreen	
This class is used to draw the Start Screen and handle the user interactions	39
pro2::TextWriter	
It is used to store the font, color palette and the charsets, as well as drawing characters onto the screen	40
pro2::Window	43

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

block.cc	51
block.hh	51
coin.cc	52
coin.hh	53
fenster.h	54
finder.hh	60
game.cc	62
game.hh	62
geometry.cc	63
geometry.hh	64
interactables.cc	66
interactables.hh	66
main.cc	67
mario.cc	68
mario.hh	68
platform.cc	70
platform.hh	70
start_screen.cc	71
start_screen.hh	71
text.cc	72
text.hh	73
utils.cc	75
utils.hh	75
window.cc	77
window.hh	77

Chapter 4

Namespace Documentation

4.1 pro2 Namespace Reference

Classes

- struct [DoubPt](#)
- struct [Pt](#)
- struct [Rect](#)
- class [TextWriter](#)

It is used to store the font, color palette and the charsets, as well as drawing characters onto the screen.

- class [Window](#)

Typedefs

- typedef std::vector< std::vector< std::vector< std::string > > > [Font](#)
Vector of characters, which each character being an array of strings. Each element of the array represents a pixel to be painted, and the string is mapped to the color it should be painted into.
- typedef std::map< std::string, int > [Palette](#)
Map from the strings to the colors.
- typedef std::map< char, int > [Charset](#)
Map from the characters to the index of the corresponding sprite from the font.
- 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)
Returns true if it has crossed the rect vertically and an integer with the direction in which it has coressed:
- `std::pair< bool, int > resolve_collision_horizontal` (const `Rect` &prev, `Rect` curr, const `Rect` &block)
Returns true if it has crossed the rect horizontally and an integer with the direction in which it has coressed:
- `bool operator<` (const `Pt` &a, const `Pt` &b)
Compares two points.
- `Pt round_dpt` (const `DoubPt` &a)
Returns a `pro2::Pt` with the truncated values of a `pro2::DoubPt`.
- `bool check_collision` (const `Rect` &a, const `Rect` &b)
Returns true if the two `pro2::Rect` are intersecting.
- `std::ifstream read_file` (std::string fname)
Loads a text file as a `stream`.
- `Palette read_colors` (std::ifstream &stream)
Generates a palette of colors from a text file.
- `Font read_sprites` (std::ifstream &stream)
Generates a `font` (vector of sprites of every character) from a text file. The font text files have the following structure:
- `Charset read_charset` (std::ifstream &stream)
Loads the charset (available characters and how they are ordered) of the font from a text file.
- `Sprite color_sprite` (const std::vector< std::vector< std::string > > &sprite, const `Palette` &colors)
Paints the character of the font with a color palette.
- `void paint_char` (`Window` &window, `Pt` &pos, `Sprite` sprite, int &size)
Draws onto the screen a character with a position and size.
- `std::vector< std::string > split_lines` (std::string text)
Divides a multiline text (separated by ' ') into a vector of each line's string.
- `double random_double` (int min, int max, int precision)
Returns a `double` between `min` and `max` with `n` digits of precision.
- `void paint_hline` (`Window` &window, int xini, int xfin, int y, `Color` color=`white`)
Draws a horizontal line onto the screen.
- `void paint_vline` (`Window` &window, int x, int yini, int yfin, `Color` color=`white`)
Draws a vertical line onto the screen.
- `void paint_rect` (`Window` &window, `Rect` rect, `Color` color, int brush_sz)
Draws a `pro2::Rect` rectangle.
- `void paint_rect_fill` (`Window` &window, `Rect` rect, `Color` color)
Draws and fills a `pro2::Rect` rectangle.
- `void paint_rect_fill_transparent` (`Window` &window, `Rect` rect, `Color` color, double transp)
Draws Dibuixa i emplena un rectangle 'Rect' amb transparència.
- `void paint_pixel_transparent` (`Window` &window, `Pt` pos, `Color` color, double transp)
Paints a pixel with a color and transparency.
- `void paint_sprite` (`Window` &window, `Pt` orig, const `Sprite` &sprite, bool mirror)
Draws a sprite onto the screen.

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
```

Map from the characters to the index of the corresponding sprite from the font.

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 of characters, which each character being an array of strings. Each element of the array represents a pixel to be painted, and the string is mapped to the color it should be painted into.

4.1.1.4 Palette

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

Map from the strings to the 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]
```

Returns true if the two [pro2::Rect](#) are intersecting.

4.1.3.2 color_sprite()

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

Paints the character of the font with a color palette.

Parameters

<i>sprite</i>	Sprite of the character with the templates of the color.
<i>colors</i>	Color palette.

4.1.3.3 operator<()

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

Compares two points.

The comparison is necessary for [Pt](#) to be used as a `map` key. It first checks the x coordinate, and if they are equal, it checks the y coordinate.

4.1.3.4 paint_char()

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

Draws onto the screen a character with a position and size.

Parameters

<i>window</i>	Window where to draw.
<i>pos</i>	Position where to draw.
<i>sprite</i>	Sprite of the painted character.
<i>size</i>	Text size.

4.1.3.5 paint_hline()

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

Draws a horizontal line onto the screen.

Parameters

<i>window</i>	Window where to draw.
<i>xini</i>	Initial x coordinate.
<i>xfin</i>	Final x coordinate.
<i>y</i>	Y coordinate.
<i>color</i>	Line color (by default it is set to white).

4.1.3.6 paint_pixel_transparent()

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

Paints a pixel with a color and transparency.

Parameters

<i>window</i>	Window where to draw.
<i>pos</i>	pro2::Pt with the position of the pixel.
<i>color</i>	Color .
<i>transp</i>	Transparency (from 0 for opaque to 1 for totally transparent)

4.1.3.7 paint_rect()

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

Draws a [pro2::Rect](#) rectangle.

Parameters

<i>window</i>	Window where to draw.
<i>rect</i>	pro2::Rect to draw.
<i>color</i>	Line color.
<i>brush_sz</i>	Line width.

4.1.3.8 paint_rect_fill()

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

Draws and fills a pro2::Rect rectangle.

Parameters

<i>window</i>	Window where to draw.
<i>rect</i>	pro2::Rect to draw.
<i>color</i>	Color of the rectangle.

4.1.3.9 paint_rect_fill_transparent()

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

Draws Dibuixa i emplena un rectangle 'Rect' amb transparència.

Parameters

<i>window</i>	Window where to draw.
<i>rect</i>	pro2::Rect to draw.
<i>color</i>	Color of the rectangle.
<i>transp</i>	Transparency (from 0 for opaque to 1 for totally transparent)

4.1.3.10 paint_sprite()

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

Draws a sprite onto the screen.

Parameters

<i>window</i>	Window where to draw.
<i>orig</i>	pro2::Pt with the upper left corner of the sprite.
<i>sprite</i>	pro2::Sprite to draw.
<i>mirror</i>	True/False to mirror the sprite.

4.1.3.11 paint_vline()

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

Draws a vertical line onto the screen.

Parameters

<i>window</i>	Window where to draw.
<i>x</i>	X coordinate.
<i>yini</i>	Initial y coordinate.
<i>yfin</i>	Final y coordinate.
<i>color</i>	Line color (by default it is set to white).

4.1.3.12 random_double()

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

Returns a double between `min` and `max` with `n` digits of precision.

Parameters

<i>min</i>	Minimum value.
<i>max</i>	Maximum value.
<i>precision</i>	Integer equal to 10^n (<code>n</code> digits of preceision).

4.1.3.13 read_charset()

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

Loads the charset (available characters and how they are ordered) of the font from a text file.

Parameters

<i>stream</i>	Stream of the text file.
---------------	--------------------------

4.1.3.14 read_colors()

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

Generates a palette of colors from a text file.

Parameters

<i>stream</i>	Stream of the text file.
---------------	--------------------------

4.1.3.15 read_file()

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

Loads a text file as a stream.

Parameters

<i>fname</i>	File path.
--------------	------------

4.1.3.16 read_sprites()

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

Generates a font (vector of sprites of every character) from a text file. The font text files have the following structure:

- On the first line, there are three integers with 'count' (number of sprites), 'height' and 'width' (of each sprites).
- The next lines contain all of the sprites with the characters of the font.

Parameters

<i>stream</i>	Stream of the text file.
---------------	--------------------------

4.1.3.17 resolve_collision_horizontal()

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

Returns true if it has crossed the rect horizontally and an integer with the direction in which it has coressed:

- 1 if it has crossed from left to right.
- -1 if it has crossed from right to left.

4.1.3.18 resolve_collision_vertical()

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

Returns true if it has crossed the rect vertically and an integer with the direction in which it has coressed:

- 1 if it has crossed from below.
- -1 if it has crossed from above.

4.1.3.19 round_dpt()

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

Returns a `pro2::Pt` with the truncated values of a `pro2::DoubPt`.

4.1.3.20 split_lines()

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

Divides a multiline text (separated by '
') into a vector of each line's string.

Parameters

<i>text</i>	Text to divide.
-------------	-----------------

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
 - Returns what type of block it is.*
- [pro2::Rect](#) get_rect () const
- std::vector< std::vector< int > > [get_sprite](#) (int anim_frame) const
 - Returns the array of the sprite for the corresponding frame of the animation.*
- int [check_bumped](#) (int state)
 - Checks collision with a block and if it needs to be activated. Returns:*

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]
```

Returns what type of block it is.

Returns

- 0 if it is a 'brick block'
- 1 if it is a 'question mark block'
- 2 if it is an 'activated block'

5.1.2.2 check_bumped()

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

Checks collision with a block and if it needs to be activated. Returns:

- 0 if nothing should happen
- 1 if the block has to be destroyed
- 2 if a coin has to be spawned (without breaking the block)
- 3 if a mushroom has to be spawned (without breaking the block)

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]
```

Returns the array of the sprite for the corresponding frame of the animation.

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

Class which implements buttons and their functionality The buttons can be clicked, and when the cursor goes over them the background color can be changed (bg_normal/bg_selected)

```
#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
Returns true if the button is being hovered.
- void [paint](#) ([pro2::Window](#) &window, [pro2::TextWriter](#) &writer) const

5.2.1 Detailed Description

Class which implements buttons and their functionality The buttons can be clicked, and when the cursor goes over them the background color can be changed (bg_normal/bg_selected)

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]
```

Returns true if the button is being hovered.

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})`
Coins are initialised with an initial position and, optionally, speed, acceleration and drag. Leaving speed, acceleration and drag on default, the coin will be spawned floating and static.
- `void paint (pro2::Window &window, int anim_frame) const`
Paints the sprite of the coin of the current animation.
- `pro2::Pt pos () const`
- `pro2::Rect get_rect () const`
- `std::vector< std::vector< int > > get_sprite (int anim_frame) const`
Returns the array of the current sprite.
- `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]
```

Coins are initialised with an initial position and, optionally, speed, acceleration and drag. Leaving speed, acceleration and drag on default, the coin will be spawned floating and static.

Parameters

<i>pos</i>	
<i>speed</i>	
<i>accel</i>	
<i>drag</i>	

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]
```

Returns the array of the current sprite.

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
```

Paints the sprite of the coin of the current animation.

Parameters

<i>window</i>	Window where it will be drawn
---------------	-------------------------------

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](#)})
Creates a [Finder<T>](#) object and initializes the container with empty cells.
- void [add](#) ([T](#) *t)
Adds an element to finder.
- void [update](#) ([T](#) *t)
Updates an object's coordinates.
- void [remove](#) ([T](#) *t)
Removes an object from the container.
- void [remove_and_delete](#) ([T](#) *t)
Removes an objet from the conainer and deletes it.
- [std::set< T * >](#) [query](#) ([pro2::Rect](#) rect) const
Returns a set with the pointers which are within the rect.
- void [AddFromList](#) ([std::list< T >](#) &set)
Adds all of the element from a list to the container.

5.6.1 Detailed Description

```
template<typename T>
class Finder< T >
```

This class consists in a container divided into cells. Each object will be added to the corresponding cell based on its coordinates.

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]
```

Creates a [Finder<T>](#) object and initializes the container with empty cells.

Precondition

The type 'T' objects must have a `get_rect` method that returns a [pro2::Rect](#)

Parameters

<i>range</i>	pro2::Rect with {xmin, ymin, xmax, ymax} of the range of coordinates it accepts
<i>divider</i>	pro2::Pt with the size of each cell

The default values are {0, 0, MAX_SZ, MAX_SZ} and {MAX_SZ/NUM_DIVS, MAX_SZ/NUM_DIVS}, with MAX_SZ = 20000 and NUM_DIVS = 32

5.6.3 Member Function Documentation

5.6.3.1 add()

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

Adds an element to finder.

Parameters

<i>t</i>	(T *) Pointer to the element to be added.
----------	---

5.6.3.2 AddFromList()

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

Adds all of the element from a list to the container.

5.6.3.3 query()

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

Returns a set with the pointers which are within the rect.

Parameters

<i>rect</i>	pro2::Rect with the coordinates to check
-------------	--

5.6.3.4 remove()

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

Removes an object from the container.

Parameters

<i>T</i>	*t Pointer to the object.
----------	---------------------------

Precondition

The object must already be present in the container.

5.6.3.5 remove_and_delete()

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

Removes an objet from the conainer and deletes it.

Parameters

<i>T</i>	*t Pointer to the object.
----------	---------------------------

Precondition

The object must already be present in the container.

5.6.3.6 update()

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

Updates an object's coordinates.

Parameters

<i>T</i>	* <i>t</i> Pointer to the object
----------	----------------------------------

Precondition

The object must already be present in the container.

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
Returns 1 if the player has died, 2 if it has won or otherwise 0.
- void [spawn_coin](#) ([pro2::Pt](#) pos, [pro2::DoubPt](#) vel)
Creates a new coin and adds it to the finder.
- void [anim_step](#) ()
Updates the animation counter to go to the next 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]
```

Updates the animation counter to go to the next sprite.

5.7.2.2 exit_code()

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

Returns 1 if the player has died, 2 if it has won or otherwise 0.

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)
```

Creates a new coin and adds it to the 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

The class [Interactable](#) is used for every type of object which interacti with [Mario](#).

```
#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

The class `Interactable` is used for every type of object which interact with `Mario`.

There are 4 types:

- 0: `Coin` which is created by a block when mario jumps into it from below. It acts as an animation, floating upwards and disappearing automatically. The coin is added to the coin counter.
- 1: Mushroom which makes `Mario` change into its 'big' state. Once it is spawned, it moves horizontally and it is also affected by gravity and block collisions. It disappears once `Mario` touches it.
- 2: The level end flag. It triggers the game win condition once `Mario` touches it.
- 3: An enemy 'goomba'. If `Mario` jumps on top of it, it is despawned and a coin is spawned. If the goomba touches `Mario` in any other way, it damages `Mario`.

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)
Adds coins to the coin counter.
- int [get_coin_count](#) ()
Returns the value of the coin counter.
- void [update](#) ([pro2::Window](#) &window, const [Finder](#)< [Platform](#) > &platforms, [Finder](#)< [Block](#) > &blocks, std::list< [Interactable](#) > &interactables)
- void [set_state](#) (int new_state)
Changes the state of the player to 'small' (0) or 'big' (1). Also the states 2 and 3 are used for the animation of changing from small to big or the other way.
- int [get_state](#) () const
Returns the state of [Mario](#): 'small' (0) or 'big' (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]
```

Adds coins to the coin counter.

Parameters

<i>ammount</i>	Ammount of coins to add.
----------------	--------------------------

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]
```

Returns the value of the coin counter.

5.9.2.4 get_state()

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

Returns the state of [Mario](#): 'small' (0) or 'big' (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)
```

Changes the state of the player to 'small' (0) or 'big' (1). Also the states 2 and 3 are used for the animation of changing from small to big or the other way.

Parameters

<i>new_state</i>	New state.
------------------	------------

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

This class is used to draw the Start Screen and handle the user interactions.

```
#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
Returns 1 if the 'play' button has been pressed and -1 if the 'quit' button has been pressed.
- void [restart](#) ([pro2::Window](#) &window)
Resets the class to its default values.

5.13.1 Detailed Description

This class is used to draw the Start Screen and handle the user interactions.

```
The Start Screen has two buttons, one to play and the other to quit.
```

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]
```

Returns 1 if the 'play' button has been pressed and -1 if the 'quit' button has been pressed.

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)
```

Resets the class to its default values.

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

It is used to store the font, color palette and the charsets, as well as drawing characters onto the screen.

```
#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
Returns the sprite of the character painted with the palette. If the character does not exist within the font, it returns a NULL character sprite (last character in the 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})
Draws a string as text on the screen. Dibuixa un string com a text a la pantalla.

5.14.1 Detailed Description

It is used to store the font, color palette and the charsets, as well as drawing characters onto the screen.

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
```

Returns the sprite of the character painted with the palette. If the character does not exist within the font, it returns a NULL character sprite (last character in the font).

Parameters

<i>ch</i>	Character
-----------	-----------

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})
```

Draws a string as text on the screen. Dibuixa un string com a text a la pantalla.

Parameters

<i>window</i>	Window where to draw.
<i>orig</i>	Origin coordinates for the textbox (depends on alignment).
<i>text</i>	String with the text to draw.
<i>space_between_chars</i>	Space that will be left between characters.
<i>size</i>	Tamany (gruix) del text
<i>alignment</i>	Determines how the text will be aligned: {x_align, y_align}. x_align and y_align can have the values [0,1,2], corresponding to [left/top, center, right/bottom] respectively. i.e. with the align {0,0}, the textbox will have at is orig the top left corner of the 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

<i>title</i>	El título de la ventana (un literal de cadena de caracteres)
<i>width</i>	El ancho de la ventana en píxels.
<i>height</i>	El alto de la ventana en píxels.
<i>zoom</i>	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

<i>color</i>	El color a utilizar para pintar. Se puede usar uno de los valores del enumerado <code>Colors</code> , como <code>red</code> , o bien poner un entero en hexadecimal, como <code>0x0084fb</code> , 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 <code>#0084fb</code> (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

<code>xy</code>	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('l')) { ... }
if (window.is_key_down(Key::Escape)) { ... }
```

Parameters

<code>code</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 <code>enum Key</code> , 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

<i>key</i>	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

<i>desplazamiento</i>	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 superficie, 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

<i>topleft</i>	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

<i>fps</i>	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, sino 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

<i>xy</i>	Coordenadas del pixel que se quiere cambiar
<i>color</i>	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

<i>ms</i>	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

<i>code</i>	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 <code>enum Key</code> , 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_; // 1 if it is a brick, 1 question mark or 2 activated block
00016         int has_object_; // 1 if it has a coin, 2 if it has a mushroom, nothing otherwise
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_; // Left upper corner new position and last position
00015     pro2::DoubPt speed_;
00016     pro2::DoubPt accel_; // By default: -gravity (floating coin)
00017     pro2::DoubPt drag_coef_; // F = -b*v
00018
00019     bool grounded_ = false;
00020
00021     void apply_physics_();
00022
00023 public:
00024
00025     Coin(pro2::Pt pos, pro2::DoubPt speed = {0, 0}, pro2::DoubPt accel = {0, -1}, pro2::DoubPt drag =
00026 {0.075, 0.075}) :
00027     pos_(pos), last_pos_(pos), speed_(speed), accel_(accel), drag_coef_(drag) {}
00028
00029     void paint(pro2::Window& window, int anim_frame) const;
00030
00031     pro2::Pt pos() const {
00032         return pos_;
00033     }
00034
00035     inline pro2::Rect get_rect() const {
00036         return {pos_.x - sz_w/2, pos_.y - sz_h, pos_.x + sz_w/2, pos_.y};
00037     }
00038
00039     inline std::vector<std::vector<int>> get_sprite(int anim_frame) const {
00040         return sprites[animation[anim_frame]];
00041     }
00042
00043     void set_y(int y) {
00044         pos_.y = y;
00045     }
00046
00047     bool is_grounded() const {
00048         return grounded_;
00049     }
00050
00051     void set_grounded(bool grounded) {
00052         grounded_ = grounded;
00053         if (grounded_) {
00054             speed_.y = 0;
00055         }
00056     }
00057 }
```

```

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;
00081 private:
00082     static const std::vector<int> animation;
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/XKLib.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), NSUIInteger, 3, NSUIInteger, 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] =
00125 {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,
00126 // clang-format on
00127
00128 FENSTER_API int fenster_loop(struct fenster *f) {
00129     msg1(void, msg(id, f->wnd, "contentView"), "setNeedsDisplay:", BOOL, YES);
00130     id ev = msg4(id, NSApp, "nextEventMatchingMask:untilDate:inMode:dequeue:", NSUIInteger,
00131         NSUIIntegerMax, id, NULL, id, NSDefaultRunLoopMode, BOOL, YES);
00132     if (!ev) {
00133         return 0;
00134     }
00135     NSUIInteger evtype = msg(NSUIInteger, ev, "type");
00136     switch (evtype) {
00137         case 1: /* NSEventTypeMouseDown */
00138             f->mouse |= 1;
00139             break;
00140         case 2: /* NSEventTypeMouseUp */
00141             f->mouse &= ~1;
00142             break;
00143         case 5:
00144         case 6: { /* NSEventTypeMouseMoved */
00145             CGPoint xy = msg(CGPoint, ev, "locationInWindow");
00146             f->x = (int)xy.x;
00147             f->y = (int)(f->height - xy.y);
00148             return 0;
00149         }
00150         case 10: /*NSEventTypeKeyDown*/
00151         case 11: /*NSEventTypeKeyUp*/: {
00152             NSUIInteger k = msg(NSUIInteger, ev, "keyCode");
00153             f->keys[k < 127 ? FENSTER_KEYCODES[k] : 0] = evtype == 10;
00154             NSUIInteger mod = msg(NSUIInteger, ev, "modifierFlags") >> 17;
00155             f->mod = (mod & 0xc) | ((mod & 1) << 1) | ((mod >> 1) & 1);
00156             return 0;
00157         }
00158     }
00159     msg1(void, NSApp, "sendEvent:", id, ev);
00160     return 0;
00161 }
00162 #elif defined(_WIN32)
00163 // clang-format off
00164 static const uint8_t FENSTER_KEYCODES[] =
00165 {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,
00166 // clang-format on
00167
00168 typedef struct BINFO {
00169     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) & 0x1fff]] = !((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] =
00260 {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_Up,24,XK_Right,19,XK_Space,32,XK_Tab,9,XK_Up,16,XK_VK01,1,XK_VK02,2,XK_VK03,3,XK_VK04,4,XK_VK05,5,XK_VK06,6,XK_VK07,7,XK_VK08,8,XK_VK09,9,XK_VK10,10,XK_VK11,11,XK_VK12,12,XK_VK13,13,XK_VK14,14,XK_VK15,15,XK_VK16,16,XK_VK17,17,XK_VK18,18,XK_VK19,19,XK_VK20,20,XK_VK21,21,XK_VK22,22,XK_VK23,23,XK_VK24,24,XK_VK25,25,XK_VK26,26,XK_VK27,27,XK_VK28,28,XK_VK29,29,XK_VK30,30,XK_VK31,31,XK_VK32,32,XK_VK33,33,XK_VK34,34,XK_VK35,35,XK_VK36,36,XK_VK37,37,XK_VK38,38,XK_VK39,39,XK_VK40,40,XK_VK41,41,XK_VK42,42,XK_VK43,43,XK_VK44,44,XK_VK45,45,XK_VK46,46,XK_VK47,47,XK_VK48,48,XK_VK49,49,XK_VK50,50,XK_VK51,51,XK_VK52,52,XK_VK53,53,XK_VK54,54,XK_VK55,55,XK_VK56,56,XK_VK57,57,XK_VK58,58,XK_VK59,59,XK_VK60,60,XK_VK61,61,XK_VK62,62,XK_VK63,63,XK_VK64,64,XK_VK65,65,XK_VK66,66,XK_VK67,67,XK_VK68,68,XK_VK69,69,XK_VK70,70,XK_VK71,71,XK_VK72,72,XK_VK73,73,XK_VK74,74,XK_VK75,75,XK_VK76,76,XK_VK77,77,XK_VK78,78,XK_VK79,79,XK_VK80,80,XK_VK81,81,XK_VK82,82,XK_VK83,83,XK_VK84,84,XK_VK85,85,XK_VK86,86,XK_VK87,87,XK_VK88,88,XK_VK89,89,XK_VK90,90,XK_VK91,91,XK_VK92,92,XK_VK93,93,XK_VK94,94,XK_VK95,95,XK_VK96,96,XK_VK97,97,XK_VK98,98,XK_VK99,99,XK_VK100,100,XK_VK101,101,XK_VK102,102,XK_VK103,103,XK_VK104,104,XK_VK105,105,XK_VK106,106,XK_VK107,107,XK_VK108,108,XK_VK109,109,XK_VK110,110,XK_VK111,111,XK_VK112,112,XK_VK113,113,XK_VK114,114,XK_VK115,115,XK_VK116,116,XK_VK117,117,XK_VK118,118,XK_VK119,119,XK_VK120,120,XK_VK121,121,XK_VK122,122,XK_VK123,123,XK_VK124,124,XK_VK125,125,XK_VK126,126,XK_VK127,127,XK_VK128,128,XK_VK129,129,XK_VK130,130,XK_VK131,131,XK_VK132,132,XK_VK133,133,XK_VK134,134,XK_VK135,135,XK_VK136,136,XK_VK137,137,XK_VK138,138,XK_VK139,139,XK_VK140,140,XK_VK141,141,XK_VK142,142,XK_VK143,143,XK_VK144,144,XK_VK145,145,XK_VK146,146,XK_VK147,147,XK_VK148,148,XK_VK149,149,XK_VK150,150,XK_VK151,151,XK_VK152,152,XK_VK153,153,XK_VK154,154,XK_VK155,155,XK_VK156,156,XK_VK157,157,XK_VK158,158,XK_VK159,159,XK_VK160,160,XK_VK161,161,XK_VK162,162,XK_VK163,163,XK_VK164,164,XK_VK165,165,XK_VK166,166,XK_VK167,167,XK_VK168,168,XK_VK169,169,XK_VK170,170,XK_VK171,171,XK_VK172,172,XK_VK173,173,XK_VK174,174,XK_VK175,175,XK_VK176,176,XK_VK177,177,XK_VK178,178,XK_VK179,179,XK_VK180,180,XK_VK181,181,XK_VK182,182,XK_VK183,183,XK_VK184,184,XK_VK185,185,XK_VK186,186,XK_VK187,187,XK_VK188,188,XK_VK189,189,XK_VK190,190,XK_VK191,191,XK_VK192,192,XK_VK193,193,XK_VK194,194,XK_VK195,195,XK_VK196,196,XK_VK197,197,XK_VK198,198,XK_VK199,199,XK_VK200,200,XK_VK201,201,XK_VK202,202,XK_VK203,203,XK_VK204,204,XK_VK205,205,XK_VK206,206,XK_VK207,207,XK_VK208,208,XK_VK209,209,XK_VK210,210,XK_VK211,211,XK_VK212,212,XK_VK213,213,XK_VK214,214,XK_VK215,215,XK_VK216,216,XK_VK217,217,XK_VK218,218,XK_VK219,219,XK_VK220,220,XK_VK221,221,XK_VK222,222,XK_VK223,223,XK_VK224,224,XK_VK225,225,XK_VK226,226,XK_VK227,227,XK_VK228,228,XK_VK229,229,XK_VK230,230,XK_VK231,231,XK_VK232,232,XK_VK233,233,XK_VK234,234,XK_VK235,235,XK_VK236,236,XK_VK237,237,XK_VK238,238,XK_VK239,239,XK_VK240,240,XK_VK241,241,XK_VK242,242,XK_VK243,243,XK_VK244,244,XK_VK245,245,XK_VK246,246,XK_VK247,247,XK_VK248,248,XK_VK249,249,XK_VK250,250,XK_VK251,251,XK_VK252,252,XK_VK253,253,XK_VK254,254,XK_VK255,255,XK_VK256,256,XK_VK257,257,XK_VK258,258,XK_VK259,259,XK_VK260,260,XK_VK261,261,XK_VK262,262,XK_VK263,263,XK_VK264,264,XK_VK265,265,XK_VK266,266,XK_VK267,267,XK_VK268,268,XK_VK269,269,XK_VK270,270,XK_VK271,271,XK_VK272,272,XK_VK273,273,XK_VK274,274,XK_VK275,275,XK_VK276,276,XK_VK277,277,XK_VK278,278,XK_VK279,279,XK_VK280,280,XK_VK281,281,XK_VK282,282,XK_VK283,283,XK_VK284,284,XK_VK285,285,XK_VK286,286,XK_VK287,287,XK_VK288,288,XK_VK289,289,XK_VK290,290,XK_VK291,291,XK_VK292,292,XK_VK293,293,XK_VK294,294,XK_VK295,295,XK_VK296,296,XK_VK297,297,XK_VK298,298,XK_VK299,299,XK_VK300,300,XK_VK301,301,XK_VK302,302,XK_VK303,303,XK_VK304,304,XK_VK305,305,XK_VK306,306,XK_VK307,307,XK_VK308,308,XK_VK309,309,XK_VK310,310,XK_VK311,311,XK_VK312,312,XK_VK313,313,XK_VK314,314,XK_VK315,315,XK_VK316,316,XK_VK317,317,XK_VK318,318,XK_VK319,319,XK_VK320,320,XK_VK321,321,XK_VK322,322,XK_VK323,323,XK_VK324,324,XK_VK325,325,XK_VK326,326,XK_VK327,327,XK_VK328,328,XK_VK329,329,XK_VK330,330,XK_VK331,331,XK_VK332,332,XK_VK333,333,XK_VK334,334,XK_VK335,335,XK_VK336,336,XK_VK337,337,XK_VK338,338,XK_VK339,339,XK_VK340,340,XK_VK341,341,XK_VK342,342,XK_VK343,343,XK_VK344,344,XK_VK345,345,XK_VK346,346,XK_VK347,347,XK_VK348,348,XK_VK349,349,XK_VK350,350,XK_VK351,351,XK_VK352,352,XK_VK353,353,XK_VK354,354,XK_VK355,355,XK_VK356,356,XK_VK357,357,XK_VK358,358,XK_VK359,359,XK_VK360,360,XK_VK361,361,XK_VK362,362,XK_VK363,363,XK_VK364,364,XK_VK365,365,XK_VK366,366,XK_VK367,367,XK_VK368,368,XK_VK369,369,XK_VK370,370,XK_VK371,371,XK_VK372,372,XK_VK373,373,XK_VK374,374,XK_VK375,375,XK_VK376,376,XK_VK377,377,XK_VK378,378,XK_VK379,379,XK_VK380,380,XK_VK381,381,XK_VK382,382,XK_VK383,383,XK_VK384,384,XK_VK385,385,XK_VK386,386,XK_VK387,387,XK_VK388,388,XK_VK389,389,XK_VK390,390,XK_VK391,391,XK_VK392,392,XK_VK393,393,XK_VK394,394,XK_VK395,395,XK_VK396,396,XK_VK397,397,XK_VK398,398,XK_VK399,399,XK_VK400,400,XK_VK401,401,XK_VK402,402,XK_VK403,403,XK_VK404,404,XK_VK405,405,XK_VK406,406,XK_VK407,407,XK_VK408,408,XK_VK409,409,XK_VK410,410,XK_VK411,411,XK_VK412,412,XK_VK413,413,XK_VK414,414,XK_VK415,415,XK_VK416,416,XK_VK417,417,XK_VK418,418,XK_VK419,419,XK_VK420,420,XK_VK421,421,XK_VK422,422,XK_VK423,423,XK_VK424,424,XK_VK425,425,XK_VK426,426,XK_VK427,427,XK_VK428,428,XK_VK429,429,XK_VK430,430,XK_VK431,431,XK_VK432,432,XK_VK433,433,XK_VK434,434,XK_VK435,435,XK_VK436,436,XK_VK437,437,XK_VK438,438,XK_VK439,439,XK_VK440,440,XK_VK441,441,XK_VK442,442,XK_VK443,443,XK_VK444,444,XK_VK445,445,XK_VK446,446,XK_VK447,447,XK_VK448,448,XK_VK449,449,XK_VK450,450,XK_VK451,451,XK_VK452,452,XK_VK453,453,XK_VK454,454,XK_VK455,455,XK_VK456,456,XK_VK457,457,XK_VK458,458,XK_VK459,459,XK_VK460,460,XK_VK461,461,XK_VK462,462,XK_VK463,463,XK_VK464,464,XK_VK465,465,XK_VK466,466,XK_VK467,467,XK_VK468,468,XK_VK469,469,XK_VK470,470,XK_VK471,471,XK_VK472,472,XK_VK473,473,XK_VK474,474,XK_VK475,475,XK_VK476,476,XK_VK477,477,XK_VK478,478,XK_VK479,479,XK_VK480,480,XK_VK481,481,XK_VK482,482,XK_VK483,483,XK_VK484,484,XK_VK485,485,XK_VK486,486,XK_VK487,487,XK_VK488,488,XK_VK489,489,XK_VK490,490,XK_VK491,491,XK_VK492,492,XK_VK493,493,XK_VK494,494,XK_VK495,495,XK_VK496,496,XK_VK497,497,XK_VK498,498,XK_VK499,499,XK_VK500,500,XK_VK501,501,XK_VK502,502,XK_VK503,503,XK_VK504,504,XK_VK505,505,XK_VK506,506,XK_VK507,507,XK_VK508,508,XK_VK509,509,XK_VK510,510,XK_VK511,511,XK_VK512,512,XK_VK513,513,XK_VK514,514,XK_VK515,515,XK_VK516,516,XK_VK517,517,XK_VK518,518,XK_VK519,519,XK_VK520,520,XK_VK521,521,XK_VK522,522,XK_VK523,523,XK_VK524,524,XK_VK525,525,XK_VK526,526,XK_VK527,527,XK_VK528,528,XK_VK529,529,XK_VK530,530,XK_VK531,531,XK_VK532,532,XK_VK533,533,XK_VK534,534,XK_VK535,535,XK_VK536,536,XK_VK537,537,XK_VK538,538,XK_VK539,539,XK_VK540,540,XK_VK541,541,XK_VK542,542,XK_VK543,543,XK_VK544,544,XK_VK545,545,XK_VK546,546,XK_VK547,547,XK_VK548,548,XK_VK549,549,XK_VK550,550,XK_VK551,551,XK_VK552,552,XK_VK553,553,XK_VK554,554,XK_VK555,555,XK_VK556,556,XK_VK557,557,XK_VK558,558,XK_VK559,559,XK_VK560,560,XK_VK561,561,XK_VK562,562,XK_VK563,563,XK_VK564,564,XK_VK565,565,XK_VK566,566,XK_VK567,567,XK_VK568,568,XK_VK569,569,XK_VK570,570,XK_VK571,571,XK_VK572,572,XK_VK573,573,XK_VK574,574,XK_VK575,575,XK_VK576,576,XK_VK577,577,XK_VK578,578,XK_VK579,579,XK_VK580,580,XK_VK581,581,XK_VK582,582,XK_VK583,583,XK_VK584,584,XK_VK585,585,XK_VK586,586,XK_VK587,587,XK_VK588,588,XK_VK589,589,XK_VK590,590,XK_VK591,591,XK_VK592,592,XK_VK593,593,XK_VK594,594,XK_VK595,595,XK_VK596,596,XK_VK597,597,XK_VK598,598,XK_VK599,599,XK_VK600,600,XK_VK601,601,XK_VK602,602,XK_VK603,603,XK_VK604,604,XK_VK605,605,XK_VK606,606,XK_VK607,607,XK_VK608,608,XK_VK609,609,XK_VK610,610,XK_VK611,611,XK_VK612,612,XK_VK613,613,XK_VK614,614,XK_VK615,615,XK_VK616,616,XK_VK617,617,XK_VK618,618,XK_VK619,619,XK_VK620,620,XK_VK621,621,XK_VK622,622,XK_VK623,623,XK_VK624,624,XK_VK625,625,XK_VK626,626,XK_VK627,627,XK_VK628,628,XK_VK629,629,XK_VK630,630,XK_VK631,631,XK_VK632,632,XK_VK633,633,XK_VK634,634,XK_VK635,635,XK_VK636,636,XK_VK637,637,XK_VK638,638,XK_VK639,639,XK_VK640,640,XK_VK641,641,XK_VK642,642,XK_VK643,643,XK_VK644,644,XK_VK645,645,XK_VK646,646,XK_VK647,647,XK_VK648,648,XK_VK649,649,XK_VK650,650,XK_VK651,651,XK_VK652,652,XK_VK653,653,XK_VK654,654,XK_VK655,655,XK_VK656,656,XK_VK657,657,XK_VK658,658,XK_VK659,659,XK_VK660,660,XK_VK661,661,XK_VK662,662,XK_VK663,663,XK_VK664,664,XK_VK665,665,XK_VK666,666,XK_VK667,667,XK_VK668,668,XK_VK669,669,XK_VK670,670,XK_VK671,671,XK_VK672,672,XK_VK673,673,XK_VK674,674,XK_VK675,675,XK_VK676,676,XK_VK677,677,XK_VK678,678,XK_VK679,679,XK_VK680,680,XK_VK681,681,XK_VK682,682,XK_VK683,683,XK_VK684,684,XK_VK685,685,XK_VK686,686,XK_VK687,687,XK_VK688,688,XK_VK689,689,XK_VK690,690,XK_VK691,691,XK_VK692,692,XK_VK693,693,XK_VK694,694,XK_VK695,695,XK_VK696,696,XK_VK697,697,XK_VK698,698,XK_VK699,699,XK_VK700,700,XK_VK701,701,XK_VK702,702,XK_VK703,703,XK_VK704,704,XK_VK705,705,XK_VK706,706,XK_VK707,707,XK_VK708,708,XK_VK709,709,XK_VK710,710,XK_VK711,711,XK_VK712,712,XK_VK713,713,XK_VK714,714,XK_VK715,715,XK_VK716,716,XK_VK717,717,XK_VK718,718,XK_VK719,719,XK_VK720,720,XK_VK721,721,XK_VK722,722,XK_VK723,723,XK_VK724,724,XK_VK725,725,XK_VK726,726,XK_VK727,727,XK_VK728,728,XK_VK729,729,XK_VK730,730,XK_VK731,731,XK_VK732,732,XK_VK733,733,XK_VK734,734,XK_VK735,735,XK_VK736,736,XK_VK737,737,XK_VK738,738,XK_VK739,739,XK_VK740,740,XK_VK741,741,XK_VK742,742,XK_VK743,743,XK_VK744,744,XK_VK745,745,XK_VK746,746,XK_VK747,747,XK_VK748,748,XK_VK749,749,XK_VK750,750,XK_VK751,751,XK_VK752,752,XK_VK753,753,XK_VK754,754,XK_VK755,755,XK_VK756,756,XK_VK757,757,XK_VK758,758,XK_VK759,759,XK_VK760,760,XK_VK761,761,XK_VK762,762,XK_VK763,763,XK_VK764,764,XK_VK765,765,XK_VK766,766,XK_VK767,767,XK_VK768,768,XK_VK769,769,XK_VK770,770,XK_VK771,771,XK_VK772,772,XK_VK773,773,XK_VK774,774,XK_VK775,775,XK_VK776,776,XK_VK777,777,XK_VK778,778,XK_VK779,779,XK_VK780,780,XK_VK781,781,XK_VK782,782,XK_VK783,783,XK_VK784,784,XK_VK785,785,XK_VK786,786,XK_VK787,787,XK_VK788,788,XK_VK789,789,XK_VK790,790,XK_VK791,791,XK_VK792,792,XK_VK793,793,XK_VK794,794,XK_VK795,795,XK_VK796,796,XK_VK797,797,XK_VK798,798,XK_VK799,799,XK_VK800,800,XK_VK801,801,XK_VK802,802,XK_VK803,803,XK_VK804,804,XK_VK805,805,XK_VK806,806,XK_VK807,807,XK_VK808,808,XK_VK809,809,XK_VK810,810,XK_VK811,811,XK_VK812,812,XK_VK813,813,XK_VK814,814,XK_VK815,815,XK_VK816,816,XK_VK817,817,XK_VK818,818,XK_VK819,819,XK_VK820,820,XK_VK821,821,XK_VK822,822,XK_VK823,823,XK_VK824,824,XK_VK825,825,XK_VK826,826,XK_VK827,827,XK_VK828,828,XK_VK829,829,XK_VK830,830,XK_VK831,831,XK_VK832,832,XK_VK833,833,XK_VK834,834,XK_VK835,835,XK_VK836,836,XK_VK837,837,XK_VK838,838,XK_VK839,839,XK_VK840,840,XK_VK841,841,XK_VK842,842,XK_VK843,843,XK_VK844,844,XK_VK845,845,XK_VK846,846,XK_VK847,847,XK_VK848,848,XK_VK849,849,XK_VK850,850,XK_VK851,851,XK_VK852,852,XK_VK853,853,XK_VK854,854,XK_VK855,855,XK_VK856,856,XK_VK857,857,XK_VK858,858,XK_VK859,859,XK_VK860,860,XK_VK861,861,XK_VK862,862,XK_VK863,863,XK_VK864,864,XK_VK865,865,XK_VK866,866,XK_VK867,867,XK_VK868,868,XK_VK869,869,XK_VK870,870,XK_VK871,871,XK_VK872,872,XK_VK873,873,XK_VK874,874,XK_VK875,875,XK_VK876,876,XK_VK877,877,XK_VK878,878,XK_VK879,879,XK_VK880,880,XK_VK881,881,XK_VK882,882,XK_VK883,883,XK_VK884,884,XK_VK885,885,XK_VK886,886,XK_VK887,887,XK_VK888,888,XK_VK889,889,XK_VK890,890,XK_VK891,891,XK_VK892,892,XK_VK893,893,XK_VK894,894,XK_VK895,895,XK_VK896,896,XK_VK897,897,XK_VK898,898,XK_VK899,899,XK_VK900,900,XK_VK901,901,XK_VK902,902,XK_VK903,903,XK_VK904,904,XK_VK905,905,XK_VK906,906,XK_VK907,907,XK_VK908,908,XK_VK909,909,XK_VK910,910,XK_VK911,911,XK_VK912,912,XK_VK913,913,XK_VK914,914,XK_VK915,915,XK_VK916,916,XK_VK917,917,XK_VK918,918,XK_VK919,919,XK_VK920,920,XK_VK921,921,XK_VK922,922,XK_VK923,923,XK_VK924,924,XK_VK925,925,XK_VK926,926,XK_VK927,927,XK_VK928,928,XK_VK929,929,XK_VK930,930,XK_VK931,931,XK_VK932,932,XK_VK933,933,XK_VK934,934,XK_VK935,935,XK_VK936,936,XK_VK937,937,XK_VK938,938,XK_VK939,939,XK_VK940,940,XK_VK941,941,XK_VK942,942,XK_VK943,943,XK_VK944,944,XK_VK945,945,XK_VK946,946,XK_VK947,947,XK_VK948,948,XK_VK949,949,XK_VK950,950,XK_VK951,951,XK_VK952,952,XK_VK953,953,XK_VK954,954,XK_VK955,955,XK_VK956,956,XK_VK957,957,XK_VK958,958,XK_VK959,959,XK_VK960,960,XK_VK961,961,XK_VK962,962,XK_VK963,963,XK_VK964,964,XK_VK965,965,XK_VK966,966,XK_VK967,967,XK_VK968,968,XK_VK969,969,XK_VK970,970,XK_VK971,971,XK_VK972,972,XK_VK973,973,XK_VK974,974,XK_VK975,975,XK_VK976,976,XK_VK977,977,XK_VK978,978,XK_VK979,979,XK_VK980,980,XK_VK981,981,XK_VK982,982,XK_VK983,983,XK_VK984,984,XK_VK985,985,XK_VK986,986,XK_VK987,987,XK_VK988,988,XK_VK989,989,XK_VK990,990,XK_VK991,991,XK_VK992,992,XK_VK993,993,XK_VK994,994,XK_VK995,995,XK_VK996,996,XK_VK997,997,XK_VK998,998,XK_VK999,999,XK_VK1000,1000,XK_VK1001,1001,XK_VK1002,1002,XK_VK1003,1003,XK_VK1004,1004,XK_VK1005,1005,XK_VK1006,1006,XK_VK1007,1007,XK_VK1008,1008,XK_VK1009,1009,XK_VK1010,1010,XK_VK1011,1011,XK_VK1012,1012,XK_VK1013,1013,XK_VK1014,1014,XK_VK1015,1015,XK_VK1016,1016,XK_VK1017,1017,XK_VK1018,1018,XK_VK1019,1019,XK_VK1020,1020,XK_VK1021,1021,XK_VK1022,1022,XK_VK1023,1023,XK_VK1024,1024,XK_VK1025,1025,XK_VK1026,1026,XK_VK1027,1027,XK_VK1028,1028,XK_VK1029,1029,XK_VK1030,1030,XK_VK1031,1031,XK_VK1032,1032,XK_VK1033,1033,XK_VK1034,1034,XK_VK1035,1035,XK_VK1036,1036,XK_VK1037,1037,XK_VK1038,1038,XK_VK1039,1039,XK_VK1040,1040,XK_VK1041,1041,XK_VK1042,1042,XK_VK1043,1043,XK_VK1044,1044,XK_VK1045,1045,XK_VK1046,1046,XK_VK1047,1047,XK_VK1048,1048,XK_VK1049,1049,XK_VK1050,1050,XK_VK1051,1051,XK_VK1052,1052,XK_VK1053,1053,XK_VK1054,1054,XK_VK1055,1055,XK_VK1056,1056,XK_VK1057,1057,XK_VK1058,1058,XK_VK1059,1059,XK_VK1060,1060,XK_VK1061,1061,XK_VK1062,1062,XK_VK1063,1063,XK_VK1064,1064,XK_VK1065,1065,XK_VK1066,1066,XK_VK1067,1067,XK_VK1068,1068,XK_VK1069,1069,XK_VK1070,1070,XK_VK1071,1071,XK_VK1072,1072,XK_VK1073,1073,XK_VK1074,1074,XK_VK1075,1075,XK_VK1076,1076,XK_VK1077,1077,XK_VK1078,1078,XK_VK1079,1079,XK_VK1080,1080,XK_VK1081,1081,XK_VK1082,1082,XK_VK1083,1083,XK_VK1084,1084,XK_VK1085,1085,XK_VK1086,1086,XK_VK1087,1087,XK_VK1088,1088,XK_VK1089,1089,XK_VK1090,1090,XK_VK1091,1091,XK_VK1092,1092,XK_VK1093,1093,XK_VK1094,1094,XK_VK1095,1095,XK_VK1096,1096,XK_VK1097,1097,XK_VK1098,1098,XK_VK1099,1099,XK_VK1100,1100,XK_VK1101,1101,XK_VK1102,1102,XK_VK1103,1103,XK_V
```

```

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 of cells with the objects
00026         std::vector< std::set<T *> > _container;
00027
00028         // Map with the pointers to the cells
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
00078     void remove(T *t) {
00079         for (std::set<int>::iterator it = _locator.find(t)->second.begin(); it !=
00080 _locator.find(t)->second.end(); it++) {
00081             _container[*it].erase(t);
00082         }
00083         _locator.erase(t);
00084     };
00085
00086     void remove_and_delete(T *t) {
00087         remove(t);
00088         delete t;
00089     }
00090
00091     std::set<T *> query(pro2::Rect rect) const {
00092         std::set<T *> result;
00093         for (int i = (rect.top - _range.top)/_divider.y; i <= (rect.bottom -
00094 _range.top)/_divider.y; i++) {
00095             for (int j = (rect.left - _range.left)/_divider.x; j <= (rect.right -
00096 _range.left)/_divider.x; j++) {
00097                 for (typename std::set<T *>::const_iterator it = _container[i*_num_divs.y +
00098 j].begin(); it != _container[i*_num_divs.y + j].end(); it++) {
00099                     pro2::Rect obj_rect = (*it)->get_rect();
00100                     if (not(rect.left > obj_rect.right or rect.right < obj_rect.left or rect.top >
00101 obj_rect.bottom or rect.bottom < obj_rect.top)) result.insert(*it);
00102                 }
00103             }
00104         }
00105         return result;
00106     };
00107
00108     void AddFromList(std::list<T> &set) {
00109         for (typename std::list<T>::const_iterator it = set.begin(); it != set.end(); it++) {
00110             add(&(*it));
00111         }
00112     }
00113 };
00114 #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
00051     int exit_code() const {return exit_code_;}
00052
00053     void spawn_coin(pro2::Pt pos, pro2::DoubPt vel);
00054
00055     void anim_step() {
00056         curr_anim_frame_++;
00057         curr_anim_frame_ %= 18;
00058     }
00059
00060 private:
00061     static constexpr int sky_blue = 0x5c94fc;
00062     static constexpr int anim_len = 10;
00063 };
00064
00065 #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)
Returns true if it has crossed the rect vertically and an integer with the direction in which it has coressed:
- `std::pair< bool, int > pro2::resolve_collision_horizontal` (const `Rect` &prev, `Rect` curr, const `Rect` &block)
Returns true if it has crossed the rect horizontally and an integer with the direction in which it has coressed:

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)
Compares two points.
- `Pt pro2::round_dpt` (const `DoubPt` &a)
Returns a `pro2::Pt` with the truncated values of a `pro2::DoubPt`.
- `bool pro2::check_collision` (const `Rect` &a, const `Rect` &b)
Returns true if the two `pro2::Rect` are intersecting.
- `std::pair< bool, int > pro2::resolve_collision_vertical` (const `Rect` &prev, `Rect` curr, const `Rect` &block)
Returns true if it has crossed the rect vertically and an integer with the direction in which it has coressed:
- `std::pair< bool, int > pro2::resolve_collision_horizontal` (const `Rect` &prev, `Rect` curr, const `Rect` &block)
Returns true if it has crossed the rect horizontally and an integer with the direction in which it has coressed:

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
00030 inline bool operator<(const Pt& a, const Pt& b) {
00031     return a.x != b.x ? a.x < b.x : a.y < b.y;
00032 }
00033
00034 // `double` coordinate points
00035 struct DoubPt {
00036     double x = 0, y = 0;
00037 };
00038
00039 inline Pt round_dpt(const DoubPt& a) {
00040     return Pt{int(a.x), int(a.y)};
00041 };
00042
00043 struct Rect {
00044     int left, top, right, bottom;
00045     int width() const {return right - left;}
00046     int height() const {return bottom - top;}
00047
00048     Rect& operator+=(const Rect& other) {
00049         left += other.left;
00050         right += other.right;
00051         top += other.top;
00052         bottom += other.bottom;
00053         return *this;
00054     }
00055
00056     Rect& operator-=(const Rect& other) {
00057         left -= other.left;
00058         right -= other.right;
00059         top -= other.top;
00060         bottom -= other.bottom;
00061         return *this;
00062     }
00063 };
00064
00065 inline bool check_collision(const Rect& a, const Rect& b) {
00066     return ((a.left <= b.left ? a.right >= b.left : a.left <= b.right) and (a.top <= b.top ? a.bottom
00067     >= b.top : a.top <= b.bottom));
00068 };
00069
00070 std::pair<bool, int> resolve_collision_vertical(const Rect& prev, Rect curr, const Rect& block);
00071
00072 std::pair<bool, int> resolve_collision_horizontal(const Rect& prev, Rect curr, const Rect& block);
00073
00074 }
00075 #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](#)

The class [Interactable](#) is used for every type of object which interact with [Mario](#).

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
00033 class Interactable {
00034     private:
00035         int type_;
00036
00037         pro2::Pt pos_;
00038         int direction = 1;
00039
00040         int despawn_timer_;
00041
00042     public:
00043         Interactable(pro2::Pt pos, int type) : pos_(pos), type_(type), despawn_timer_((type==0)? 10 :
00044 -1) {};
00045
00046         bool update(pro2::Window& window, const Finder<Platform>& platforms, const Finder<Block>&
00047 blocks);
00048
00049         void paint(pro2::Window& window) const;
00050
00051         pro2::Rect collision_box() const {
00052             if (type_ == 0) return {0,0,0,0};
00053             else if (type_ == 1 or type_ == 3) return {pos_.x, pos_.y, pos_.x + 15, pos_.y + 15};
00054             else return {pos_.x+7, pos_.y, pos_.x + 8, pos_.y + 168};
00055         }
00056
00057         int type() const {return type_;}
00058
00059         pro2::Pt pos() const {return pos_;}
00060
00061     private:
00062         static const std::vector<std::vector<std::vector<int>>> sprites;
00063 };
00064 #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)
Draws the death screen which lasts for 5 seconds or until the 'Escape' key is pressed.
- void `win_screen` (`pro2::Window` &window, `pro2::TextWriter` &tw)
Draws the win screen which lasts for 5 seconds or until the 'Escape' key is pressed.
- 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)
```

Draws the death screen which lasts for 5 seconds or until the 'Escape' key is pressed.

6.20.1.2 `main()`

```
int main ()
```

6.20.1.3 `win_screen()`

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

Draws the win screen which lasts for 5 seconds or until the 'Escape' key is pressed.

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
00066     void add_coin(int ammount = 1) {
00067         coin_count_ += ammount;
00068     }
00069
00070     int get_coin_count() {
00071         return coin_count_;
00072     }
00073
00074     void update(pro2::Window& window, const Finder<Platform>& platforms, Finder<Block>& blocks,
00075         std::list<Interactable>& interactables);
00076
00077     void set_state(int new_state);
00078
00079     int get_state() const {return state_;}
00080
00081 private:
00082     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](#)

Class which implements buttons and their functionality The buttons can be clicked, and when the cursor goes over them the background color can be changed (bg_normal/bg_selected)

- class [StartScreen](#)

This class is used to draw the Start Screen and handle the user interactions.

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
00016 class Button {
00017     private:
00018         pro2::Rect rect;
00019         std::string text;
00020         int bg_normal, bg_selected;

```

```

00021     public:
00022         Button(
00023             pro2::Rect rect,
00024             std::string text,
00025             int bg_normal=0xCD612E,
00026             int bg_selected=0xC97A55
00027         ) : rect(rect), text(text), bg_normal(bg_normal), bg_selected(bg_selected) {};
00028
00029         pro2::Rect get_rect() const {return rect;}
00030
00031         bool selected(pro2::Pt pos) const {return (pos.x >= rect.left and pos.x <= rect.right and
00032 pos.y >= rect.top and pos.y <= rect.bottom);}
00033
00034         void paint(pro2::Window& window, pro2::TextWriter& writer) const;
00035     };
00036
00037 class StartScreen {
00038     private:
00039         std::vector< Button > buttons;
00040         pro2::TextWriter TW_;
00041
00042         // Flags
00043         bool finished_ = false;
00044         int exit_code_ = 0;
00045
00046         int height_;
00047         int width_;
00048     public:
00049         StartScreen(int width, int height, pro2::TextWriter TW);
00050
00051         inline bool is_finished() const {return finished_;}
00052
00053         void process_keys(pro2::Window& window);
00054
00055         void update(pro2::Window& window);
00056
00057         void paint(pro2::Window& window);
00058
00059         int exit_code() const {return exit_code_;}
00060
00061         void restart(pro2::Window& window);
00062     private:
00063         static constexpr int sky_blue = 0x5c94fc;
00064 };
00065 #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)

- Loads a text file as a `stream`.*
- `Palette pro2::read_colors` (`std::ifstream &stream`)
 - Generates a palette of colors from a text file.*
- `Font pro2::read_sprites` (`std::ifstream &stream`)
 - Generates a `font` (vector of sprites of every character) from a text file. The font text files have the following structure:*
- `Charset pro2::read_charset` (`std::ifstream &stream`)
 - Loads the charset (available characters and how they are ordered) of the font from a text file.*
- `Sprite pro2::color_sprite` (`const std::vector< std::vector< std::string > > &sprite`, `const Palette &colors`)
 - Paints the character of the font with a color palette.*
- `void pro2::paint_char` (`Window &window`, `Pt &pos`, `Sprite sprite`, `int &size`)
 - Draws onto the screen a character with a position and size.*
- `std::vector< std::string > pro2::split_lines` (`std::string text`)
 - Divides a multiline text (separated by ' ') into a vector of each line's string.*

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`
 - It is used to store the font, color palette and the charsets, as well as drawing characters onto the screen.*

Namespaces

- namespace `pro2`

Typedefs

- `typedef std::vector< std::vector< std::vector< std::string > > > pro2::Font`
 - Vector of characters, which each character being an array of strings. Each element of the array represents a pixel to be painted, and the string is mapped to the color it should be painted into.*
- `typedef std::map< std::string, int > pro2::Palette`
 - Map from the strings to the colors.*
- `typedef std::map< char, int > pro2::Charset`
 - Map from the characters to the index of the corresponding sprite from the font.*

Functions

- `std::ifstream` [pro2::read_file](#) (`std::string` fname)
Loads a text file as a stream.
- [Palette](#) `pro2::read_colors` (`std::ifstream` &stream)
Generates a palette of colors from a text file.
- [Font](#) `pro2::read_sprites` (`std::ifstream` &stream)
Generates a font (vector of sprites of every character) from a text file. The font text files have the following structure:
- [Charset](#) `pro2::read_charset` (`std::ifstream` &stream)
Loads the charset (available characters and how they are ordered) of the font from a text file.
- [Sprite](#) `pro2::color_sprite` (`const std::vector< std::vector< std::string > >` &sprite, `const` [Palette](#) &colors)
Paints the character of the font with a color palette.
- `void` `pro2::paint_char` ([Window](#) &window, [Pt](#) &pos, [Sprite](#) sprite, `int` &size)
Draws onto the screen a character with a position and size.
- `std::vector< std::string >` `pro2::split_lines` (`std::string` text)
Divides a multiline text (separated by ' ') into a vector of each line's string.

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 {
00013     typedef std::vector<std::vector<std::vector<std::string>>> Font;
00014
00015     typedef std::map<std::string, int> Palette;
00016     typedef std::map<char, int> Charset;
00017
00018     std::ifstream read_file(std::string fname);
00019
00020     Palette read_colors(std::ifstream &stream);
00021
00022     Font read_sprites(std::ifstream &stream);
00023
00024     Charset read_charset(std::ifstream &stream);
00025
00026     Sprite color_sprite(const std::vector<std::vector<std::string>> &sprite, const Palette &colors);
00027
00028     void paint_char(Window& window, Pt& pos, Sprite sprite, int& size);
00029
00030     std::vector<std::string> split_lines(std::string text);
00031
00032     class TextWriter {
00033     private:
00034         Font font_;
00035         Palette palette_;
00036         Charset charset_;
00037
00038     public:
00039         TextWriter(Font font, Palette palette) : font_(font), palette_(palette) {};
00040         TextWriter(Font font, std::string palette_path);
00041         TextWriter(std::string font_path, Palette palette);
00042         TextWriter(std::string font_path, std::string palette_path);
00043
00044         Sprite get_sprite(char ch) const;
00045
00046         void set_font(Font font) {font_ = font;}
00047     };
00048 }

```

```

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)
Returns a double between min and max with n digits of precision.
- void [pro2::paint_hline](#) (Window &window, int xini, int xfin, int y, [Color](#) color=[white](#))
Draws a horizontal line onto the screen.
- void [pro2::paint_vline](#) (Window &window, int x, int yini, int yfin, [Color](#) color=[white](#))
Draws a vertical line onto the screen.
- void [pro2::paint_rect](#) (Window &window, [Rect](#) rect, [Color](#) color, int brush_sz)
Draws a [pro2::Rect](#) rectangle.
- void [pro2::paint_rect_fill](#) (Window &window, [Rect](#) rect, [Color](#) color)
Draws and fills a [pro2::Rect](#) rectangle.
- void [pro2::paint_rect_fill_transparent](#) (Window &window, [Rect](#) rect, [Color](#) color, double transp)
Draws Dibuixa i emplena un rectangle 'Rect' amb transparència.
- void [pro2::paint_pixel_transparent](#) (Window &window, [Pt](#) pos, [Color](#) color, double transp)
Paints a pixel with a color and transparency.
- void [pro2::paint_sprite](#) (Window &window, [Pt](#) orig, const [Sprite](#) &sprite, bool mirror)
Draws a sprite onto the screen.

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)
Returns a double between min and max with n digits of precision.
- void [pro2::paint_hline](#) (Window &>window, int xini, int xfin, int y, [Color](#) color=[white](#))
Draws a horizontal line onto the screen.
- void [pro2::paint_vline](#) (Window &>window, int x, int yini, int yfin, [Color](#) color=[white](#))
Draws a vertical line onto the screen.
- void [pro2::paint_rect](#) (Window &>window, [Rect](#) rect, [Color](#) color, int brush_sz)
Draws a [pro2::Rect](#) rectangle.
- void [pro2::paint_rect_fill](#) (Window &>window, [Rect](#) rect, [Color](#) color)
Draws and fills a [pro2::Rect](#) rectangle.
- void [pro2::paint_rect_fill_transparent](#) (Window &>window, [Rect](#) rect, [Color](#) color, double transp)
Draws Dibuixa i emplena un rectangle 'Rect' amb transparència.
- void [pro2::paint_pixel_transparent](#) (Window &>window, [Pt](#) pos, [Color](#) color, double transp)
Paints a pixel with a color and transparency.
- void [pro2::paint_sprite](#) (Window &>window, [Pt](#) orig, const [Sprite](#) &sprite, bool mirror)
Draws a sprite onto the screen.

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>           // For RNG
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  
00095 void paint_pixel_transparent(  
00096     Window& window,  
00097     Pt      pos,  
00098     Color   color,  
00099     double  transp  
00100 );  
00101  
00110 void paint_sprite(Window&      window,  
00111                    Pt          orig,  
00112                    const Sprite& sprite,  
00113                    bool         mirror);  
00114 }  
00115  
00116 // namespace pro2  
00117  
00118 #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:
00072     int last_keys_[256];
00073     int last_mouse_;
00074     fenster fenster_;
00075
00076     uint32_t *pixels_;
00077
00078     size_t pixels_size_;
00079
00080     int zoom_ = 1;
00081
00082     int64_t last_time_;
00083
00084     int frame_count_ = 0;
00085
00086     uint8_t fps_ = 60;
00087
00088     // Cámara
00089
00090     Pt topleft_ = {0, 0};
00091     Pt topleft_target_ = {0, 0};
00092
00093     void update_camera_();
00094
00095     bool camera_moving() const {
00096         return topleft_.x != topleft_target_.x || topleft_.y != topleft_target_.y;
00097     }
00098
00099     static constexpr int camera_speed_ = 8;
00100
00101 public:
00102     Window(std::string title, int width, int height, int zoom = 1);
00103
00104     ~Window() {
00105         fenster_.close(&fenster_);
00106         delete[] pixels_;
00107     }
00108
00109     int width() const {
00110         return fenster_.width / zoom_;
00111     }
00112
00113     int height() const {
00114         return fenster_.height / zoom_;
00115     }
00116
00117     bool next_frame();
00118
00119     void clear(Color color = black);
00120
00121     int frame_count() const {
00122         return frame_count_;
00123     }
00124
00125     bool is_key_down(int code) const {
00126         return code >= 0 && code < 128 && fenster_.keys[code];
00127     }
00128
00129     bool was_key_pressed(int code) const {
00130         return code >= 0 && code < 128 && !last_keys_[code] && fenster_.keys[code];
00131     }
00132
00133     bool is_modkey_down(ModKey key) const {
00134         return fenster_.mod & uint8_t(key);
00135     }

```

```

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

- `_DEFAULT_SOURCE`
 - `fenster.h`, [54](#)
 - `~Window`
 - `pro2::Window`, [44](#)
- `add`
 - `Finder< T >`, [26](#)
- `add_coin`
 - `Mario`, [33](#)
- `AddFromList`
 - `Finder< T >`, [26](#)
- `Alt`
 - `pro2`, [10](#)
- `anim_step`
 - `Game`, [29](#)
- `Backspace`
 - `pro2`, [9](#)
- `black`
 - `pro2`, [16](#)
- `Block`, [17](#)
 - `Block`, [17](#)
 - `block_type`, [18](#)
 - `check_bumped`, [18](#)
 - `get_rect`, [18](#)
 - `get_sprite`, [18](#)
 - `paint`, [18](#)
 - `pos`, [19](#)
 - `sprites`, [19](#)
- `block.cc`, [51](#)
- `block.hh`, [51](#)
- `block_type`
 - `Block`, [18](#)
- `blue`
 - `pro2`, [16](#)
- `bottom`
 - `pro2::Rect`, [38](#)
- `buf`
 - `fenster`, [24](#)
- `Button`, [19](#)
 - `Button`, [20](#)
 - `get_rect`, [20](#)
 - `paint`, [20](#)
 - `selected`, [20](#)
- `camera_center`
 - `pro2::Window`, [45](#)
- `camera_rect`
 - `pro2::Window`, [45](#)
- `Charset`
 - `pro2`, [9](#)
- `check_bumped`
 - `Block`, [18](#)
- `check_collision`
 - `pro2`, [10](#)
- `clear`
 - `pro2::Window`, [45](#)
- `Coin`, [20](#)
 - `Coin`, [21](#)
 - `get_rect`, [21](#)
 - `get_sprite`, [21](#)
 - `is_grounded`, [22](#)
 - `paint`, [22](#)
 - `pos`, [22](#)
 - `set_grounded`, [22](#)
 - `set_y`, [22](#)
 - `sprites`, [23](#)
 - `toggle_grounded`, [22](#)
 - `update`, [22](#)
- `coin.cc`, [52](#)
- `coin.hh`, [53](#)
- `collision_box`
 - `Interactable`, [31](#)
 - `Mario`, [33](#)
- `Color`
 - `platform.cc`, [70](#)
 - `pro2`, [9](#)
- `color_sprite`
 - `pro2`, [10](#)
- `Ctrl`
 - `pro2`, [10](#)
- `cyan`
 - `pro2`, [16](#)
- `death_screen`
 - `main.cc`, [67](#)
- `Delete`
 - `pro2`, [9](#)
- `DIVIDER`
 - `finder.hh`, [61](#)
- `Down`
 - `pro2`, [9](#)
- `dpy`
 - `fenster`, [24](#)
- `End`
 - `pro2`, [9](#)
- `Escape`
 - `pro2`, [9](#)
- `exit_code`

- Game, 29
- StartScreen, 39
- fenster, 23
 - buf, 24
 - dpy, 24
 - gc, 24
 - height, 24
 - img, 24
 - keys, 24
 - mod, 24
 - mouse, 24
 - title, 25
 - w, 25
 - width, 25
 - x, 25
 - y, 25
- fenster.h, 54
 - _DEFAULT_SOURCE, 54
 - FENSTER_API, 54
 - fenster_close, 55
 - fenster_loop, 55
 - fenster_open, 55
 - fenster_pixel, 55
 - fenster_sleep, 55
 - fenster_time, 55
- FENSTER_API
 - fenster.h, 54
- fenster_close
 - fenster.h, 55
- FENSTER_HEADER
 - window.hh, 78
- fenster_loop
 - fenster.h, 55
- fenster_open
 - fenster.h, 55
- fenster_pixel
 - fenster.h, 55
- fenster_sleep
 - fenster.h, 55
- fenster_time
 - fenster.h, 55
- Finder
 - Finder< T >, 26
- Finder< T >, 25
 - add, 26
 - AddFromList, 26
 - Finder, 26
 - query, 27
 - remove, 27
 - remove_and_delete, 27
 - update, 27
- finder.hh, 60
 - DIVIDER, 61
 - MAX_SZ, 61
 - NUM_DIVS, 61
- Font
 - pro2, 9
- FPS
 - main.cc, 68
- frame_count
 - pro2::Window, 45
- Game, 29
 - anim_step, 29
 - exit_code, 29
 - Game, 29
 - is_finished, 30
 - is_paused, 30
 - paint, 30
 - spawn_coin, 30
 - update, 30
- game.cc, 62
- game.hh, 62
- gc
 - fenster, 24
- geometry.cc, 63
- geometry.hh, 64
- get_charset
 - pro2::TextWriter, 41
- get_coin_count
 - Mario, 33
- get_font
 - pro2::TextWriter, 41
- get_palette
 - pro2::TextWriter, 41
- get_pixel
 - pro2::Window, 45
- get_rect
 - Block, 18
 - Button, 20
 - Coin, 21
 - Platform, 35
- get_sprite
 - Block, 18
 - Coin, 21
 - pro2::TextWriter, 41
- get_state
 - Mario, 33
- green
 - pro2, 16
- has_crossed_floor_downwards
 - Platform, 35
- HEIGHT
 - main.cc, 68
- height
 - fenster, 24
 - pro2::Rect, 38
 - pro2::Window, 46
- Home
 - pro2, 9
- img
 - fenster, 24
- Insert
 - pro2, 9
- Interactable, 30

- collision_box, 31
- Interactable, 31
- paint, 31
- pos, 31
- type, 31
- update, 32
- interactables.cc, 66
- interactables.hh, 66
- is_finished
 - Game, 30
 - StartScreen, 39
- is_grounded
 - Coin, 22
 - Mario, 33
- is_key_down
 - pro2::Window, 46
- is_modkey_down
 - pro2::Window, 46
- is_mouse_down
 - pro2::Window, 47
- is_paused
 - Game, 30
- is_pt_inside
 - Platform, 35
- jump
 - Mario, 33
- Keys
 - pro2, 9
- keys
 - fenster, 24
- Left
 - pro2, 9
- left
 - pro2::Rect, 38
- magenta
 - pro2, 16
- main
 - main.cc, 67
- main.cc, 67
 - death_screen, 67
 - FPS, 68
 - HEIGHT, 68
 - main, 67
 - WIDTH, 68
 - win_screen, 67
 - ZOOM, 68
- Mario, 32
 - add_coin, 33
 - collision_box, 33
 - get_coin_count, 33
 - get_state, 33
 - is_grounded, 33
 - jump, 33
 - Mario, 33
 - paint, 33
 - pos, 34
 - set_grounded, 34
 - set_state, 34
 - set_y, 34
 - toggle_grounded, 34
 - update, 34
- mario.cc, 68
- mario.hh, 68
- MAX_SZ
 - finder.hh, 61
- Meta
 - pro2, 10
- mod
 - fenster, 24
- ModKey
 - pro2, 9
- mouse
 - fenster, 24
- mouse_pos
 - pro2::Window, 47
- move_camera
 - pro2::Window, 47
- next_frame
 - pro2::Window, 47
- NUM_DIVS
 - finder.hh, 61
- operator<
 - pro2, 10
- operator+
 - pro2::Pt, 36
- operator+=
 - pro2::Pt, 36
 - pro2::Rect, 38
- operator-
 - pro2::Pt, 37
- operator-=
 - pro2::Pt, 37
 - pro2::Rect, 38
- PageDown
 - pro2, 9
- PageUp
 - pro2, 9
- paint
 - Block, 18
 - Button, 20
 - Coin, 22
 - Game, 30
 - Interactable, 31
 - Mario, 33
 - Platform, 36
 - StartScreen, 39
- paint_char
 - pro2, 10
- paint_hline
 - pro2, 11
- paint_pixel_transparent

- pro2, 11
- paint_rect
 - pro2, 11
- paint_rect_fill
 - pro2, 12
- paint_rect_fill_transparent
 - pro2, 12
- paint_sprite
 - pro2, 12
- paint_vline
 - pro2, 13
- Palette
 - pro2, 9
- Platform, 35
 - get_rect, 35
 - has_crossed_floor_downwards, 35
 - is_pt_inside, 35
 - paint, 36
 - Platform, 35
 - top, 36
- platform.cc, 70
 - Color, 70
- platform.hh, 70
- pos
 - Block, 19
 - Coin, 22
 - Interactable, 31
 - Mario, 34
- pro2, 7
 - Alt, 10
 - Backspace, 9
 - black, 16
 - blue, 16
 - Charset, 9
 - check_collision, 10
 - Color, 9
 - color_sprite, 10
 - Ctrl, 10
 - cyan, 16
 - Delete, 9
 - Down, 9
 - End, 9
 - Escape, 9
 - Font, 9
 - green, 16
 - Home, 9
 - Insert, 9
 - Keys, 9
 - Left, 9
 - magenta, 16
 - Meta, 10
 - ModKey, 9
 - operator<, 10
 - PageDown, 9
 - PageUp, 9
 - paint_char, 10
 - paint_hline, 11
 - paint_pixel_transparent, 11
 - paint_rect, 11
 - paint_rect_fill, 12
 - paint_rect_fill_transparent, 12
 - paint_sprite, 12
 - paint_vline, 13
 - Palette, 9
 - random_double, 13
 - read_charset, 13
 - read_colors, 14
 - read_file, 14
 - read_sprites, 14
 - red, 16
 - resolve_collision_horizontal, 14
 - resolve_collision_vertical, 15
 - Return, 9
 - Right, 9
 - round_dpt, 15
 - Shift, 10
 - Space, 9
 - split_lines, 15
 - Sprite, 9
 - Tab, 9
 - Up, 9
 - white, 16
 - yellow, 16
- pro2::DoubPt, 23
 - x, 23
 - y, 23
- pro2::Pt, 36
 - operator+, 36
 - operator+=", 36
 - operator-, 37
 - operator-=", 37
 - x, 37
 - y, 37
- pro2::Rect, 37
 - bottom, 38
 - height, 38
 - left, 38
 - operator+=", 38
 - operator-=", 38
 - right, 38
 - top, 38
 - width, 38
- pro2::TextWriter, 40
 - get_charset, 41
 - get_font, 41
 - get_palette, 41
 - get_sprite, 41
 - set_charset, 42
 - set_font, 42
 - set_palette, 42
 - TextWriter, 41
 - write_text, 42
- pro2::Window, 43
 - ~Window, 44
 - camera_center, 45
 - camera_rect, 45

- clear, [45](#)
- frame_count, [45](#)
- get_pixel, [45](#)
- height, [46](#)
- is_key_down, [46](#)
- is_modkey_down, [46](#)
- is_mouse_down, [47](#)
- mouse_pos, [47](#)
- move_camera, [47](#)
- next_frame, [47](#)
- set_camera_topleft, [48](#)
- set_fps, [48](#)
- set_pixel, [49](#)
- sleep, [49](#)
- topleft, [49](#)
- was_key_pressed, [49](#)
- was_mouse_pressed, [50](#)
- width, [50](#)
- Window, [44](#)
- process_keys
 - StartScreen, [40](#)
- query
 - Finder< T >, [27](#)
- random_double
 - pro2, [13](#)
- read_charset
 - pro2, [13](#)
- read_colors
 - pro2, [14](#)
- read_file
 - pro2, [14](#)
- read_sprites
 - pro2, [14](#)
- red
 - pro2, [16](#)
- remove
 - Finder< T >, [27](#)
- remove_and_delete
 - Finder< T >, [27](#)
- resolve_collision_horizontal
 - pro2, [14](#)
- resolve_collision_vertical
 - pro2, [15](#)
- restart
 - StartScreen, [40](#)
- Return
 - pro2, [9](#)
- Right
 - pro2, [9](#)
- right
 - pro2::Rect, [38](#)
- round_dpt
 - pro2, [15](#)
- selected
 - Button, [20](#)
- set_camera_topleft
 - pro2::Window, [48](#)
- set_charset
 - pro2::TextWriter, [42](#)
- set_font
 - pro2::TextWriter, [42](#)
- set_fps
 - pro2::Window, [48](#)
- set_grounded
 - Coin, [22](#)
 - Mario, [34](#)
- set_palette
 - pro2::TextWriter, [42](#)
- set_pixel
 - pro2::Window, [49](#)
- set_state
 - Mario, [34](#)
- set_y
 - Coin, [22](#)
 - Mario, [34](#)
- Shift
 - pro2, [10](#)
- sleep
 - pro2::Window, [49](#)
- Space
 - pro2, [9](#)
- spawn_coin
 - Game, [30](#)
- split_lines
 - pro2, [15](#)
- Sprite
 - pro2, [9](#)
- sprites
 - Block, [19](#)
 - Coin, [23](#)
- start_screen.cc, [71](#)
- start_screen.hh, [71](#)
- StartScreen, [39](#)
 - exit_code, [39](#)
 - is_finished, [39](#)
 - paint, [39](#)
 - process_keys, [40](#)
 - restart, [40](#)
 - StartScreen, [39](#)
 - update, [40](#)
- Tab
 - pro2, [9](#)
- text.cc, [72](#)
- text.hh, [73](#)
- TextWriter
 - pro2::TextWriter, [41](#)
- title
 - fenster, [25](#)
- toggle_grounded
 - Coin, [22](#)
 - Mario, [34](#)
- top
 - Platform, [36](#)
 - pro2::Rect, [38](#)

- toleft
 - pro2::Window, [49](#)
- type
 - Interactable, [31](#)
- Up
 - pro2, [9](#)
- update
 - Coin, [22](#)
 - Finder< T >, [27](#)
 - Game, [30](#)
 - Interactable, [32](#)
 - Mario, [34](#)
 - StartScreen, [40](#)
- utils.cc, [75](#)
- utils.hh, [75](#)
- w
 - fenster, [25](#)
- was_key_pressed
 - pro2::Window, [49](#)
- was_mouse_pressed
 - pro2::Window, [50](#)
- white
 - pro2, [16](#)
- WIDTH
 - main.cc, [68](#)
- width
 - fenster, [25](#)
 - pro2::Rect, [38](#)
 - pro2::Window, [50](#)
- win_screen
 - main.cc, [67](#)
- Window
 - pro2::Window, [44](#)
- window.cc, [77](#)
- window.hh, [77](#)
 - FENSTER_HEADER, [78](#)
- write_text
 - pro2::TextWriter, [42](#)
- x
 - fenster, [25](#)
 - pro2::DoubPt, [23](#)
 - pro2::Pt, [37](#)
- y
 - fenster, [25](#)
 - pro2::DoubPt, [23](#)
 - pro2::Pt, [37](#)
- yellow
 - pro2, [16](#)
- ZOOM
 - main.cc, [68](#)