

jun 12, 18 14:03

GirderSize.cpp

Page 1/1

```

1  #include "GirderSize.h"
2  #include "Math.h"
3  #include "ObjectSizes.h"
4
5  float GirderSize::getGirderWidthMeters(int size, int angle){
6      angle = GirderSize::normalizeAngle(angle);
7      return Math::cosDegrees(angle) * size +
8             Math::sinDegrees(angle) * girder_height;
9  }
10
11 int GirderSize::getGirderWidthPixels(int size, int angle){
12     return SCALE_FACTOR * GirderSize::getGirderWidthMeters(size, angle);
13 }
14
15 float GirderSize::getGirderHeightMeters(int size, int angle){
16     angle = GirderSize::normalizeAngle(angle);
17     return Math::sinDegrees(angle) * size +
18            Math::cosDegrees(angle) * girder_height;
19 }
20
21 int GirderSize::getGirderHeightPixels(int size, int angle){
22     return SCALE_FACTOR * GirderSize::getGirderHeightMeters(size, angle);
23 }
24
25 int GirderSize::normalizeAngle(int angle){
26     return angle > 90 ? 180 - angle : angle;
27 }

```

jun 04, 18 21:12

GirderSize.h

Page 1/1

```

1  #ifndef __GIRDERSIZE_H__
2  #define __GIRDERSIZE_H__
3
4  class GirderSize{
5      private:
6          //Normaliza el angulo entre 0 y 90
7          static int normalizeAngle(int angle);
8      public:
9          //Devuelve el ancho de una viga en metros
10         static float getGirderWidthMeters(int size, int angle);
11
12         //Devuelve el ancho de una viga en pixeles
13         static int getGirderWidthPixels(int size, int angle);
14
15         //Devuelve el alto de una viga en metros
16         static float getGirderHeightMeters(int size, int angle);
17
18         //Devuelve el alto de una viga en pixeles
19         static int getGirderHeightPixels(int size, int angle);
20 };
21
22 #endif
23

```

jun 12, 18 14:03

Position.cpp

Page 1/1

```

1  #include "Position.h"
2  #include <cmath>
3
4  #define FACTOR 100
5
6  Position::Position(float x, float y): x(x), y(y){}
7
8  Position::~~Position(){}
9
10 bool Position::operator==(const Position& other){
11     return (int)(this->x * FACTOR) == (int)(other.x * FACTOR) &&
12            (int)(this->y * FACTOR) == (int)(other.y * FACTOR);
13 }
14
15 float Position::getX() const{
16     return this->x;
17 }
18
19 float Position::getY() const{
20     return this->y;
21 }

```

jun 12, 18 14:03

Position.h

Page 1/1

```

1  #ifndef __POSITION_H__
2  #define __POSITION_H__
3
4  /* Clase que se encarga de representar posiciones en el plano */
5  class Position{
6      private:
7          float x;
8          float y;
9
10     public:
11         /* Constructor */
12         Position(float x, float y);
13
14         /* Destructor */
15         ~Position();
16
17         /* Devuelve true si las dos posiciones son iguales */
18         bool operator==(const Position& other);
19
20         /* Devuelve el valor en X de la posicion */
21         float getX() const;
22
23         /* Devuelve el valor en Y de la posicion */
24         float getY() const;
25     };
26
27 #endif

```

jun 19, 18 12:35

ScrollHandler.cpp

Page 1/2

```

1  #include "ScrollHandler.h"
2  #include <gtkmm/adjustment.h>
3  #include <glibmm/main.h>
4
5  #define SPACE_TO_SCROLL 20
6  #define SCROLL_INCREMENT 25
7  #define TIMEOUT 50
8  #define WAIT_TO_SCROLL TIMEOUT * 10
9  const Position NO_SCROLL_POSITION(SPACE_TO_SCROLL * 2, SPACE_TO_SCROLL * 2);
10
11 ScrollHandler::ScrollHandler(Gtk::ScrolledWindow& window):
12     window(window),
13     last_mouse_position(NO_SCROLL_POSITION),
14     mouse_in_window(false){
15     this->current_time = 0;
16     this->window.add_events(Gdk::POINTER_MOTION_MASK);
17     this->window.add_events(Gdk::ENTER_NOTIFY_MASK);
18     this->window.add_events(Gdk::ENTER_NOTIFY_MASK);
19     this->window.signal_motion_notify_event().connect(
20         sigc::mem_fun(*this, &ScrollHandler:
21 :mouseMotionEvent));
22     this->window.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_NEVER);
23
24     this->window.signal_enter_notify_event().connect(
25         sigc::mem_fun(*this, &ScrollHandler:
26 :mouseEntered));
27     this->window.signal_leave_notify_event().connect(
28         sigc::mem_fun(*this, &ScrollHandler:
29 :mouseLeft));
30     this->my_connection = Glib::signal_timeout().connect(
31         sigc::mem_fun(*this, &ScrollHandler:
32 :scroll), TIMEOUT);
33 }
34
35 ScrollHandler::~~ScrollHandler(){}
36
37 bool ScrollHandler::mouseMotionEvent(GdkEventMotion* motion_event){
38     this->last_mouse_position = Position(motion_event->x, motion_event->y);
39     this->mouse_in_window = true;
40     return true;
41 }
42
43 bool ScrollHandler::mouseEntered(GdkEventCrossing* crossing_event){
44     this->mouse_in_window = true;
45     this->current_time = 0;
46     this->last_mouse_position = NO_SCROLL_POSITION;
47     return true;
48 }
49
50 bool ScrollHandler::mouseLeft(GdkEventCrossing* crossing_event){
51     this->mouse_in_window = false;
52     this->current_time = 0;
53     return true;
54 }
55
56 bool ScrollHandler::scroll(){
57     int window_width = window.get_hadjustment()->get_page_size();
58     int window_height = window.get_vadjustment()->get_page_size();
59
60     if (!this->mouse_in_window){
61         //El mouse esta fuera de la pantalla
62         this->current_time = 0;
63         return true;
64     }
65
66     if (this->current_time < WAIT_TO_SCROLL){

```

jun 19, 18 12:35

ScrollHandler.cpp

Page 2/2

```

63     this->current_time += TIMEOUT;
64     return true;
65 }
66
67 int scrolled = 0;
68 if (last_mouse_position.getX() < SPACE_TO_SCROLL){
69     //Scroll a la izquierda
70     this->window.get_hadjustment()->set_value(
71         this->window.get_hadjustment()->get_value() - SCROLL_INC
72 REMENT);
73     scrolled++;
74 }
75
76 if (last_mouse_position.getX() > window_width - SPACE_TO_SCROLL){
77     //Scroll a la derecha
78     this->window.get_hadjustment()->set_value(
79         this->window.get_hadjustment()->get_value() + SCROLL_INC
80 REMENT);
81     scrolled++;
82 }
83
84 if (last_mouse_position.getY() < SPACE_TO_SCROLL){
85     //Scroll arriba
86     this->window.get_vadjustment()->set_value(
87         this->window.get_vadjustment()->get_value() - SCROLL_INC
88 REMENT);
89     scrolled++;
90 }
91
92 if (last_mouse_position.getY() > window_height - SPACE_TO_SCROLL){
93     //Scroll abajo
94     this->window.get_vadjustment()->set_value(
95         this->window.get_vadjustment()->get_value() + SCROLL_INC
96 REMENT);
97     scrolled++;
98 }
99
100 if (!scrolled){
101     this->current_time = 0;
102 }
103 return true;
104 }
105
106 void ScrollHandler::stop(){
107     if (this->my_connection.connected()) {
108         this->my_connection.disconnect();
109     }
110 }

```

jun 12, 18 19:03

ScrollHandler.h

Page 1/1

```

1  #ifndef __SCROLLHANDLER_H__
2  #define __SCROLLHANDLER_H__
3
4  #include <gtkmm/scrolledwindow.h>
5  #include <gdk/gdk.h>
6  #include "Position.h"
7
8  class ScrollHandler{
9  private:
10     Gtk::ScrolledWindow& window;
11     Position last_mouse_position;
12     bool mouse_in_window;
13     sigc::connection my_connection;
14     int current_time;
15
16     /* Handler del movimiento del mouse */
17     bool mouseMotionEvent(GdkEventMotion* motion_event);
18
19     /* Handler de entrada en el area de desplazamiento */
20     bool mouseEntered(GdkEventCrossing* crossing_event);
21
22     /* Handler de salida del area de desplazamiento */
23     bool mouseLeft(GdkEventCrossing* crossing_event);
24
25     /* Realiza el desplazamiento de la pantalla */
26     bool scroll();
27
28 public:
29     /* Constructor */
30     explicit ScrollHandler(Gtk::ScrolledWindow& window);
31
32     /* Destructor */
33     ~ScrollHandler();
34
35     /* Detiene el desplazamiento */
36     void stop();
37 };
38
39 #endif

```

jun 12, 18 14:03

ViewPositionTransformer.cpp

Page 1/1

```

1  #include "ViewPositionTransformer.h"
2  #include "ObjectSizes.h"
3
4  ViewPositionTransformer::ViewPositionTransformer(Gtk::Layout& layout):
5      layout(layout){}
6
7  ViewPositionTransformer::~ViewPositionTransformer() {}
8
9  Position ViewPositionTransformer::transformToScreen(const Position& position){
10     guint width, height;
11     this->layout.get_size(width, height);
12     float x = SCALE_FACTOR * position.getX();
13     float y = height - SCALE_FACTOR * position.getY();
14     return Position(x, y);
15 }
16
17 Position ViewPositionTransformer::transformToScreenAndMove(
18     const Position& position, float width, float height)
19 {
20     Position pos = this->transformToScreen(position);
21     Position moved(pos.getX() - SCALE_FACTOR * width / 2,
22                  pos.getY() - SCALE_FACTOR * height / 2);
23     return moved;
24 }
25
26 Position ViewPositionTransformer::transformToPosition(const Position& position){
27     guint width, height;
28     this->layout.get_size(width, height);
29     float x = position.getX() / SCALE_FACTOR;
30     float y = (height - position.getY()) / (SCALE_FACTOR);
31     return Position(x, y);
32 }

```

jun 12, 18 14:03

ViewPositionTransformer.h

Page 1/1

```

1  #ifndef __VIEWTRANSFORMER_H__
2  #define __VIEWTRANSFORMER_H__
3
4  #include <gtkmm/layout.h>
5  #include "Position.h"
6
7  /* Clase que se encarga de transformar posiciones de la pantalla
8   * en posiciones en metros */
9  class ViewPositionTransformer{
10     private:
11         Gtk::Layout& layout;
12
13     public:
14         /* Constructor */
15         explicit ViewPositionTransformer(Gtk::Layout& layout);
16
17         /* Destructor */
18         ~ViewPositionTransformer();
19
20         /* Dada una posicion en metros, devuelve una posicion en
21          * pixeles que representa una posicion de la pantalla*/
22         Position transformToScreen(const Position& position);
23
24         /* Dada una posicion en metros, la transforma en una posicion
25          * para la pantalla y la desplaza segun su ancho y alto */
26         Position transformToScreenAndMove(const Position& pos, float w, float h)
27         ;
28
29         /* Dada una posicion en pixeles, devuelve una posicion en metros */
30         Position transformToPosition(const Position& position);
31     };
32 #endif

```

may 31, 18 21:43

Water.cpp

Page 1/1

```

1  #include "Water.h"
2  #include "Path.h"
3  #include "ObjectSizes.h"
4
5  Water::Water() {}
6
7  Water::~Water() {}
8
9  void Water::show(Gtk::Layout& layout){
10     this->images.clear();
11
12     size_t pos = 0;
13     guint width, height;
14     layout.get_size(width, height);
15
16     while(pos < width){
17         Gtk::Image image;
18         image.set(IMAGES_PATH + "Water.png");
19         this->images.push_back(std::move(image));
20         layout.put(this->images.back(), pos, height - water_height);
21         this->images.back().show();
22         pos += water_length;
23     }
24 }

```

jun 12, 18 14:03	Water.h	Page 1/1
1	<code>#ifndef __WATER_H__</code>	
2	<code>#define __WATER_H__</code>	
3		
4	<code>#include <gtkmm/image.h></code>	
5	<code>#include <gtkmm/layout.h></code>	
6	<code>#include <vector></code>	
7		
8	<code>/* Clase que se encarga de controlar la vista del agua */</code>	
9	<code>class Water{</code>	
10	<code>private:</code>	
11	<code>std::vector<Gtk::Image> images;</code>	
12		
13	<code>public:</code>	
14	<code>/* Constructor */</code>	
15	<code>Water();</code>	
16		
17	<code>/* Destructor */</code>	
18	<code>~Water();</code>	
19		
20	<code>/* Muestra la imagen del agua */</code>	
21	<code>void show(Gtk::Layout& layout);</code>	
22	<code>};</code>	
23		
24	<code>#endif</code>	

jun 26, 18 14:00	Table of Content	Page 1/1
1	Table of Contents	
2	1 <i>GirderSize.cpp</i> sheets 1 to 1 (1) pages 1- 1 28 lines	
3	2 <i>GirderSize.h</i> sheets 1 to 1 (1) pages 2- 2 24 lines	
4	3 <i>Position.cpp</i> sheets 2 to 2 (1) pages 3- 3 22 lines	
5	4 <i>Position.h</i> sheets 2 to 2 (1) pages 4- 4 28 lines	
6	5 <i>ScrollHandler.cpp</i> ... sheets 3 to 3 (1) pages 5- 6 107 lines	
7	6 <i>ScrollHandler.h</i> sheets 4 to 4 (1) pages 7- 7 40 lines	
8	7 <i>ViewPositionTransformer.cpp</i> sheets 4 to 4 (1) pages 8- 8 32 lines	
9	8 <i>ViewPositionTransformer.h</i> sheets 5 to 5 (1) pages 9- 9 33 lines	
10	9 <i>Water.cpp</i> sheets 5 to 5 (1) pages 10- 10 25 lines	
11	10 <i>Water.h</i> sheets 6 to 6 (1) pages 11- 11 25 lines	