```cpp
#include "GirderSize.h"
#include "Math.h"
#include "ObjectSizes.h"

float GirderSize::getGirderWidthMeters(int size, int angle){
    angle = GirderSize::normalizeAngle(angle);
    return Math::cosDegrees(angle) * size +
            Math::sinDegrees(angle) * girder_height;
}

int GirderSize::getGirderWidthPixels(int size, int angle){
    return SCALE_FACTOR * GirderSize::getGirderWidthMeters(size, angle);
}

float GirderSize::getGirderHeightMeters(int size, int angle){
    angle = GirderSize::normalizeAngle(angle);
    return Math::sinDegrees(angle) * size +
            Math::cosDegrees(angle) * girder_height;
}

int GirderSize::getGirderHeightPixels(int size, int angle){
    return SCALE_FACTOR * GirderSize::getGirderHeightMeters(size, angle);
}

int GirderSize::normalizeAngle(int angle){
    return angle > 90 ? 180 - angle : angle;
}
```

```cpp
#ifndef __GIRDERSIZE_H__
#define __GIRDERSIZE_H__

class GirderSize{
    private:
        //Normaliza el angulo entre 0 y 90
        static int normalizeAngle(int angle);
    public:
        //Devuelve el ancho de una viga en metros
        static float getGirderWidthMeters(int size, int angle);

        //Devuelve el ancho de una viga en pixeles
        static int getGirderWidthPixels(int size, int angle);

        //Devuelve el alto de una viga en metros
        static float getGirderHeightMeters(int size, int angle);

        //Devuelve el alto de una viga en pixeles
        static int getGirderHeightPixels(int size, int angle);
};

#endif
```

```cpp
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  }
```

```cpp
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
```

```cpp
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:
    :mouseMotionEvent));
21      this->window.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_NEVER);
22
23      this->window.signal_enter_notify_event().connect(
24                                          sigc::mem_fun(*this, &ScrollHandler:
    :mouseEntered));
25      this->window.signal_leave_notify_event().connect(
26                                          sigc::mem_fun(*this, &ScrollHandler:
    :mouseLeft));
27      this->my_connection = Glib::signal_timeout().connect(
28                                          sigc::mem_fun(*this, &ScrollHandler:
    :scroll), TIMEOUT);
29  }
30
31  ScrollHandler::~ScrollHandler(){}
32
33  bool ScrollHandler::mouseMotionEvent(GdkEventMotion* motion_event){
34      this->last_mouse_position = Position(motion_event->x, motion_event->y);
35      this->mouse_in_window = true;
36      return true;
37  }
38
39  bool ScrollHandler::mouseEntered(GdkEventCrossing* crossing_event){
40      this->mouse_in_window = true;
41      this->current_time = 0;
42      this->last_mouse_position = NO_SCROLL_POSITION;
43      return true;
44  }
45
46  bool ScrollHandler::mouseLeft(GdkEventCrossing* crossing_event){
47      this->mouse_in_window = false;
48      this->current_time = 0;
49      return true;
50  }
51
52  bool ScrollHandler::scroll(){
53      int window_width = window.get_hadjustment()->get_page_size();
54      int window_height = window.get_vadjustment()->get_page_size();
55
56      if (!this->mouse_in_window){
57          //El mouse esta fuera de la pantalla
58          this->current_time = 0;
59          return true;
60      }
61
62      if (this->current_time < WAIT_TO_SCROLL){
```

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

```
1   #ifndef __SCROLLHADNLER_H__
2   #define __SCROLLHADNLER_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
```

```
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   }
```

```cpp
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          /* Dada una posicion en pixeles, devuelve una posicion en metros */
29          Position transformToPosition(const Position& position);
30  };
31
32  #endif
```

```cpp
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  }
```

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