```
1
2   #include <Path.h>
3   #include "FileBoxController.h"
4   #include "FileWriter.h"
5   #include "FileReader.h"
6   #include "InvalidMapError.h"
7
8   static const char *const NEW_FILE_NAME = "Sin titulo.yaml";
9
10  FileBoxController::FileBoxController(UsablesController &wep_controller,
11          std::shared_ptr<MapController> map_controller,
12          const Glib::RefPtr<Gtk::Builder> &builder )
13          : usables_controller(wep_controller),
14            map_controller(std::move(map_controller))
15  {
16      builder->get_widget("save_dialog",save_dialog);
17      save_dialog->add_button("Cancelar", Gtk::RESPONSE_CANCEL);
18      save_dialog->add_button("Guardar", Gtk::RESPONSE_OK);
19
20      builder->get_widget("map_name",map_name);
21
22      builder->get_widget("open_dialog",open_dialog);
23      open_dialog->add_button("Cancelar", Gtk::RESPONSE_CANCEL);
24      open_dialog->add_button("Abrir", Gtk::RESPONSE_OK);
25
26  }
27
28  void FileBoxController::onSaveClicked() const {
29      try {
30          std::vector<std::vector<double>> worms;
31          std::vector<std::vector<double>> girders;
32          map_controller->getObjects(worms, girders);
33          Glib::RefPtr<const Gdk::Pixbuf> background = map_controller->getBackgrou
    nd();
34
35          std::vector<int> weapons_ammo;
36          unsigned int life;
37          usables_controller.getWeaponsAndLife(weapons_ammo, life);
38
39          save_dialog->set_current_folder(MAPS_PATH);
40          save_dialog->set_current_name(map_name->get_text());
41          int result = save_dialog->run();
42          if (result==Gtk::RESPONSE_OK){
43              std::string path = save_dialog->get_filename();
44              std::string filename = save_dialog->get_current_name();
45              map_name->set_label(filename);
46
47              size_t extension = filename.rfind(".");
48              std::string background_name = filename.substr(0, extension) + ".png";
49              background->save(BACKGROUND_PATH + background_name, "png");
50
51              FileWriter file(path);
52              file.save(weapons_ammo, worms,
53                        girders, life, background_name);
54          }
55          save_dialog->hide();
56
57      } catch(const InvalidMapError &error){
58          error.what();
59      }
60  }
61
62  void FileBoxController::onLoadClicked() const {
63      open_dialog->set_current_folder(MAPS_PATH);
64      int result = open_dialog->run();
65      if (result==Gtk::RESPONSE_OK) {
```



```
66          std::string filename = open_dialog->get_filename();
67          map_name->set_label(open_dialog->get_current_name());
68
69          std::vector<std::vector<double>> worms;
70          std::vector<std::vector<double>> girders;
71          std::vector<int> weps_ammo;
72          unsigned int life;
73          std::string background;
74
75          FileReader file(filename);
76          file.read(worms, girders,
77                    weps_ammo, life, background);
78
79          map_controller->loadBackground(background);
80          usables_controller.loadWeapons(weps_ammo, life);
81          map_controller->loadObjects(worms, girders);
82      }
83      open_dialog->hide();
84  }
85
86
87  void FileBoxController::onNewClicked() const {
88      map_name->set_label(NEW_FILE_NAME);
89      usables_controller.onResetSignal();
90      map_controller->newMapSignal();
91  }
92
```

Jun 08, 18 13:12 **FileBoxController.h** Page 1/1

```
1
2  #ifndef WORMS_FILECONTROLLER_H
3  #define WORMS_FILECONTROLLER_H
4
5  #include <gtkmm/filechooserdialog.h>
6  #include "FileBoxView.h"
7  #include "UsablesController.h"
8  #include "MapController.h"
9
10 // Clase que se encarga de establecer una conexion entre la seccion de archivos
11 // y  el resto del programa
12 class FileBoxController {
13 private:
14     UsablesController &usables_controller;
15     std::shared_ptr<MapController> map_controller;
16     Gtk::FileChooserDialog* save_dialog;
17     Gtk::FileChooserDialog* open_dialog;
18     Gtk::Label* map_name;
19
20 public:
21     FileBoxController(UsablesController &wep_controller,
22                       std::shared_ptr<MapController> map_controller,
23                       const Glib::RefPtr<Gtk::Builder> &builder);
24
25     // Se encarga de mostrar un cuadro de dialogo para seleccionar un archivo
26     // cuando se eligio guardar en la vista
27     void onSaveClicked() const;
28
29     // Se encarga de mostrar un cuadro de dialogo para seleccionar un archivo
30     // cuando se eligio cargar en la vista
31     void onLoadClicked() const;
32
33     // Crea un nuevo mapa y actualiza la informacion del nombre del mapa actual
34     void onNewClicked() const;
35 };
36
37
38 #endif //WORMS_FILECONTROLLER_H
```

Jun 09, 18 16:29 **MapController.cpp** Page 1/3

```
1
2  #include <gtkmm/messagedialog.h>
3  #include <ViewPositionTransformer.h>
4  #include "MapController.h"
5  #include "InvalidMapError.h"
6  #include "Path.h"
7
8  #define ADD_MODE_ID 0
9  #define MOVE_CMD_ID 1
10 #define SELECT_MODE_ID 2
11
12
13 MapController::MapController(Map model,
14                               const Glib::RefPtr<Gtk::Builder> &builder)
15         : model(std::move(model)), item_id_to_add(1),
16           actual_mode(ADD_MODE_ID)
17 {
18     builder->get_widget_derived("map", view);
19     builder->get_widget_derived("toolbox", toolBox);
20     view->bindController(this);
21     toolBox->bindController(this);
22
23     builder->get_widget("background_dialog",background_dialog);
24     background_dialog->add_button("Cancelar", Gtk::RESPONSE_CANCEL);
25     background_dialog->add_button("Abrir", Gtk::RESPONSE_OK);
26 }
27
28 void MapController::addModeSignal(const unsigned int &id) {
29     this->actual_mode = ADD_MODE_ID;
30     this->item_id_to_add = id;
31 }
32
33 void MapController::eraseSignal() {
34     model.erase(index_object_selected);
35     view->erase(index_object_selected);
36     toolBox->hideSelected();
37     toolBox->disableMovingItems();
38 }
39
40 void MapController::newMapSignal() {
41     model.clean();
42     view->clean();
43     toolBox->closeSelectionMode();
44 }
45
46 void MapController::moveSignal() {
47     this->actual_mode = MOVE_CMD_ID;
48 }
49
50 void MapController::changeModeSignal() {
51     this->actual_mode = (actual_mode==ADD_MODE_ID? SELECT_MODE_ID:ADD_MODE_ID);
52     if (actual_mode==ADD_MODE_ID) toolBox->closeSelectionMode();
53 }
54
55 void MapController::turn(const int &rotation) {
56     if (model.isGirder(index_object_selected)) {
57         unsigned int id;
58         int new_angle = this->model.turn(index_object_selected, id, rotation);
59         this->view->turn(id, new_angle, index_object_selected);
60     }
61 }
62
63 void MapController::turnCCWSignal()  {
64     turn(10);
65 }
66
```

```
67  void MapController::turnCWSignal()  {
68      turn(-10);
69  }
70
71  void MapController::mapClickedSignal(GdkEventButton *event_button) {
72      if (actual_mode == MOVE_CMD_ID) {
73          this->model.move(index_object_selected, event_button->x,
74                           event_button->y);
75          this->view->move(index_object_selected, event_button->x,
76                           event_button->y);
77          actual_mode = SELECT_MODE_ID;
78      } else if (actual_mode == SELECT_MODE_ID) {
79          this->index_object_selected = view->select(event_button->x,
80                                                     event_button->y);
81          if (index_object_selected > -1) {
82              toolBox->enableMovingItems();
83              toolBox->showSelected(model.getItemID(index_object_selected));
84          } else {
85              toolBox->disableMovingItems();
86              toolBox->hideSelected();
87          }
88          actual_mode = SELECT_MODE_ID; //cambio de estado del toolbox llama a add
     mode
89      } else {
90          this->model.add(item_id_to_add, event_button->x, event_button->y);
91          this->view->add(item_id_to_add, event_button->x, event_button->y);
92      }
93  }
94
95  void MapController::getObjects(std::vector<std::vector<double>> &worms,
96                                 std::vector<std::vector<double>> &girders) const
    {
97      model.getObjects(worms, girders);
98      if (worms.empty()){
99          throw InvalidMapError("El mapa actual no contiene worms");
100     }
101     if (girders.empty()){
102         throw InvalidMapError("El mapa actual no contiene vigas");
103     }
104
105     ViewPositionTransformer transformer(*view);
106     for (std::vector<double> &worm : worms){
107         Position position(worm[0],worm[1]);
108         Position new_pos = transformer.transformToPosition(position);
109         worm[0] = new_pos.getX();
110         worm[1] = new_pos.getY();
111     }
112     for (std::vector<double> &girder : girders){
113         Position position(girder[1],girder[2]);
114         Position new_pos = transformer.transformToPosition(position);
115         girder[1] = new_pos.getX();
116         girder[2] = new_pos.getY();
117     }
118 }
119
120 void MapController::loadObjects(std::vector<std::vector<double>> &worms,
121                                 std::vector<std::vector<double>> &girders) {
122     newMapSignal();
123     ViewPositionTransformer transformer(*view);
124     for (std::vector<double> &worm:worms) {
125         Position position(worm[0],worm[1]);
126         Position new_pos = transformer.transformToScreen(position);
127         worm[0] = new_pos.getX();
128         worm[1] = new_pos.getY();
129         this->model.add(1, worm[0], worm[1]);
130         this->view->add(1, worm[0], worm[1]);
```



```
131     }
132     for (std::vector<double> &girder:girders) {
133         Position position(girder[1],girder[2]);
134         Position new_pos = transformer.transformToScreen(position);
135         girder[1] = new_pos.getX();
136         girder[2] = new_pos.getY();
137         this->model.add(girder[0], girder[1], girder[2], girder[3]);
138         this->view->add(girder[0], girder[1], girder[2], girder[3]);
139     }
140 }
141
142 void MapController::changeBackgroundSignal() const {
143     this->background_dialog->set_current_folder(BACKGROUND_PATH);
144     int result = this->background_dialog->run();
145     if (result==Gtk::RESPONSE_OK) {
146         std::string path = this->background_dialog->get_filename();
147         this->view->changeBackground(path);
148     }
149     this->background_dialog->hide();
150 }
151
152 Glib::RefPtr<const Gdk::Pixbuf> MapController::getBackground() const{
153     return view->getBackground();
154 }
155
156 void MapController::loadBackground(const std::string &background)  {
157     view->loadBackground(background);
158 }
```

Jun 09, 18 16:24 **MapController.h** Page 1/1

```
1
2  #ifndef WORMS_MAPCONTROLLER_H
3  #define WORMS_MAPCONTROLLER_H
4
5  #include <gtkmm/filechooserdialog.h>
6  #include "MapView.h"
7  #include "Map.h"
8  #include "ToolBoxView.h"
9
10 class MapView;
11 class ToolBoxView;
12
13 // Clase que se encarga de comunicar la vista con el modelo, y a su vez, se
14 // comunica con el resto del programa
15 class MapController {
16     Map model;
17     MapView *view;
18     ToolBoxView *toolBox;
19     unsigned int item_id_to_add;
20     unsigned int actual_mode;
21     int index_object_selected;
22     Gtk::FileChooserDialog* background_dialog;
23
24     void turn(const int &rotation);
25 public:
26     MapController(Map model, const Glib::RefPtr<Gtk::Builder> &builder);
27
28     void addModeSignal(const unsigned int &id);
29
30     void eraseSignal();
31
32     void newMapSignal();
33
34     void moveSignal();
35
36     void turnCCWSignal();
37
38     void mapClickedSignal(GdkEventButton *event_button);
39
40     void getObjects(std::vector<std::vector<double>> &worms,
41                     std::vector<std::vector<double>> &girders) const;
42
43     void loadObjects(std::vector<std::vector<double>> &worms,
44                      std::vector<std::vector<double>> &girders);
45
46     void turnCWSignal();
47
48     void changeBackgroundSignal() const;
49
50     void changeModeSignal();
51
52     Glib::RefPtr<const Gdk::Pixbuf> getBackground() const;
53
54     void loadBackground(const std::string &background);
55 };
56
57
58 #endif //WORMS_MAPCONTROLLER_H
```

Jun 05, 18 14:07 **UsablesController.cpp** Page 1/2

```
1
2  #include "UsablesController.h"
3  #include "InvalidMapError.h"
4
5  UsablesController::UsablesController(
6          const Glib::RefPtr<Gtk::Builder> &builder) {
7      builder->get_widget("btn_reset", reset_button);
8      reset_button->signal_clicked().connect(
9              sigc::mem_fun(*this,
10                           &UsablesController::onResetSignal));
11
12     builder->get_widget_derived("life", life_spinner);
13
14     for (size_t i = 1; i <= 10; ++i) {
15         std::shared_ptr<WeaponView> weapon_view(new WeaponView(builder, i));
16
17         std::shared_ptr<Weapon> weapon
18                 (new Weapon(weapon_view->getInitialAmmo()));
19
20         weapons.push_back(weapon);
21
22         std::shared_ptr<WeaponController> weapon_controller(
23                 new WeaponController(weapon_view,
24                                      weapon));
25         wep_controllers.push_back(std::move(weapon_controller));
26         weapons_view.push_back(weapon_view);
27     }
28 }
29
30 void UsablesController::onResetSignal() {
31     life_spinner->reset();
32     for (const std::shared_ptr<WeaponController> &actual_controller:wep_controll
   ers) {
33         actual_controller->resetAmmo();
34     }
35 }
36
37 void UsablesController::getWeaponsAndLife(std::vector<int> &weps_ammo,
38                                           unsigned int &life) const {
39     life = life_spinner->get_value();
40     for (const std::shared_ptr<WeaponController> &actual_controller:wep_controll
   ers) {
41         weps_ammo.push_back(actual_controller->getAmmo());
42     }
43     if (!isValidWeaponSet(weps_ammo)){
44         throw InvalidMapError("NingÃºn arma tiene municiÃ³n");
45     }
46 }
47
48 void UsablesController::loadWeapons(std::vector<int> &weps_ammo,
49                                     const unsigned int &life) const {
50     int i = 0;
51     for (const std::shared_ptr<WeaponController> &actual_controller
52             :wep_controllers) {
53         actual_controller->updateAmmo(weps_ammo[i]);
54         i++;
55     }
56     life_spinner->update(life);
57 }
58
59 bool
60 UsablesController::isValidWeaponSet(std::vector<int> &ammo_vector) const {
61     for (int actual_ammo : ammo_vector) {
62         if(actual_ammo !=0)
63             return true;
64     }
```

Jun 05, 18 14:07 **UsablesController.cpp** Page 2/2

```
65      return false;
66  }
```

Jun 08, 18 13:12 **UsablesController.h** Page 1/1

```
1
2   #ifndef WORMS_WEAPONSLISTCONTROLLER_H
3   #define WORMS_WEAPONSLISTCONTROLLER_H
4
5
6   #include <gtkmm/button.h>
7   #include <gtkmm/spinbutton.h>
8   #include "Weapon.h"
9   #include "WeaponView.h"
10  #include "LifeView.h"
11
12  // Clase que se encaga de manejar la comunicacion de la vida y el arma con las
13  // demas partes del programa
14  class UsablesController {
15  private:
16      LifeView *life_spinner;
17      Gtk::Button *reset_button;
18      std::vector<std::shared_ptr<Weapon>> weapons;
19      std::vector<std::shared_ptr<WeaponView>> weapons_view;
20      std::vector<std::shared_ptr<WeaponController> > wep_controllers;
21
22      // Indica si el set actual de armas es valido (alguno con municion positiva)
23      bool isValidWeaponSet(std::vector<int> &ammo_vector) const;
24  public:
25      explicit UsablesController(
26              const Glib::RefPtr<Gtk::Builder> &builder);
27
28      // Indica a los controladores de armas y vida que deben reiniciarse
29      void onResetSignal();
30
31      // Obtiene a los valores actuales de las armas y la vida
32      void getWeaponsAndLife(std::vector<int> &weps_ammo, unsigned int &life) cons
    t;
33
34      // Establece los valores de las armas y la vida
35      void
36      loadWeapons(std::vector<int> &weps_ammo, const unsigned int &life) const;
37
38
39  };
40
41
42  #endif //WORMS_WEAPONSLISTCONTROLLER_H
```

Jun 05, 18 14:07 **WeaponController.cpp** Page 1/1

```cpp

#include "WeaponController.h"

WeaponController::WeaponController(std::shared_ptr<WeaponView> View,
                                   std::shared_ptr<Weapon> model)
        : weapon_view(std::move(View)),
          weapon_model(std::move(model)) {
    weapon_view->bindController(this);
}

void WeaponController::resetAmmo() {
    weapon_view->resetAmmo();
    weapon_model->resetAmmo();
}

void WeaponController::updateAmmo(const int &ammo) {
    weapon_model->setAmmo(ammo);
    weapon_view->setAmmo(ammo);
}

int WeaponController::getAmmo() {
    return weapon_model->getAmmo();
}
```

Jun 08, 18 13:12 **WeaponController.h** Page 1/1

```cpp

#ifndef WORMS_WEAPONCONTROLLER_H
#define WORMS_WEAPONCONTROLLER_H

#include "WeaponView.h"
#include "Weapon.h"
class WeaponView;

// Clase que se encarga de manejar la informacion del arma entre el modelo
// y la vista
class WeaponController {
private:
    std::shared_ptr<WeaponView> weapon_view;
    std::shared_ptr<Weapon> weapon_model;
public:
    WeaponController(std::shared_ptr<WeaponView>,
                     std::shared_ptr<Weapon>
                     model);

    // Indica a la vista y al modelo que deben resetear la municion
    void resetAmmo();

    // Indica a la vista y al modelo que deben establecer un nuevo valor de
    // municion especificado
    void updateAmmo(const int &ammo);

    // Obtiene el valor de la municion desde el modelo
    int getAmmo();
};


#endif //WORMS_WEAPONCONTROLLER_H
```

Jun 01, 18 13:12 **main.cpp** Page 1/1

```cpp
#include <gtkmm/application.h>
#include <gtkmm/builder.h>
#include <giomm.h>
#include <iostream>
#include <gtkmm/scrolledwindow.h>
#include <gtkmm/window.h>
#include "Editor.h"
#include "Path.h"

int main() {
    Glib::RefPtr<Gtk::Application> app = Gtk::Application::create();
    Glib::RefPtr<Gtk::Builder> refBuilder = Gtk::Builder::create();
    try {
        refBuilder->add_from_file(GLADE_PATH+"editor.glade");
    }
    catch (const Glib::FileError &ex) {
        std::cerr << "FileError: " << ex.what() << std::endl;
        return 1;
    }
    catch (const Glib::MarkupError &ex) {
        std::cerr << "MarkupError: " << ex.what() << std::endl;
        return 1;
    }
    catch (const Gtk::BuilderError &ex) {
        std::cerr << "BuilderError: " << ex.what() << std::endl;
        return 1;
    }

    Editor *mainWindow = nullptr;
    refBuilder->get_widget_derived("main_window", mainWindow);
    if (mainWindow) {
        mainWindow->set_title(EDITOR_WINDOW_NAME);
        mainWindow->set_icon_from_file(ICON_PATH);
        app->run(*mainWindow);
        delete mainWindow;
    }
    return 0;
}
```

Jun 08, 18 13:12 **Editor.cpp** Page 1/1

```cpp

#include "Editor.h"

Editor::Editor(BaseObjectType *cobject,
               const Glib::RefPtr<Gtk::Builder> &builder)
        : Gtk::Window(cobject),
          usables_controller(builder) {

    maximize();
    builder->get_widget("map_window", map_window);

    std::shared_ptr<MapController> map_controller
            (new MapController(map_model, builder));

    builder->get_widget_derived("filebox", filebox);
    std::shared_ptr<FileBoxController> filebox_controller(
            new FileBoxController(usables_controller, map_controller,builder));
    filebox->bindController(filebox_controller);

    show_all_children();
}
```

Jun 08, 18 13:12 **Editor.h** Page 1/1

```
1
2  #ifndef WORMS_EDITOR_H
3  #define WORMS_EDITOR_H
4
5  #include <gtkmm/builder.h>
6  #include <gtkmm/window.h>
7  #include <gtkmm/scrolledwindow.h>
8  #include <gtkmm/spinbutton.h>
9  #include "MapView.h"
10 #include "ToolBoxView.h"
11 #include "UsablesController.h"
12 #include "FileBoxController.h"
13 #include "FileBoxView.h"
14
15 class Editor : public Gtk::Window {
16     Gtk::ScrolledWindow *map_window;
17     Map map_model;
18     UsablesController usables_controller;
19     FileBoxView *filebox;
20
21 public:
22     Editor(BaseObjectType *cobject, const Glib::RefPtr<Gtk::Builder> &builder);
23
24 };
25
26
27 #endif //WORMS_EDITOR_H
```

Jun 02, 18 18:24 **FileReader.cpp** Page 1/1

```
1
2  #include "FileReader.h"
3
4  FileReader::FileReader(const std::string &filename)
5          : file(filename, std::fstream::in),
6            filename(filename) {}
7
8  void FileReader::read(std::vector<std::vector<double>> &worms,
9                        std::vector<std::vector<double>> &girders,
10                       std::vector<int> &weps_ammo,
11                       unsigned int &worms_life, std::string& background) {
12     YAML::Node config = YAML::LoadFile(filename);
13
14     background = config[BACKGROUND_IMAGE].as<std::string>();
15
16     worms_life = config[WORMS_LIFE].as<unsigned int>();
17
18     std::map<std::string, int> ammo = config[WEAPON_AMMO].as<std::map<std::strin
   g,
19              int>>();
20
21     weps_ammo.push_back(ammo[BAZOOKA_NAME]);
22     weps_ammo.push_back(ammo[MORTAR_NAME]);
23     weps_ammo.push_back(ammo[GREEN_GRENADE_NAME]);
24     weps_ammo.push_back(ammo[RED_GRENADE_NAME]);
25     weps_ammo.push_back(ammo[BANANA_NAME]);
26     weps_ammo.push_back(ammo[AIR_ATTACK_NAME]);
27     weps_ammo.push_back(ammo[BAT_NAME]);
28     weps_ammo.push_back(ammo[TELEPORT_NAME]);
29     weps_ammo.push_back(ammo[DYNAMITE_NAME]);
30     weps_ammo.push_back(ammo[HOLY_GRENADE_NAME]);
31
32     worms = config[WORMS_DATA].as<std::vector<std::vector<double>>>();
33
34     girders = config[GIRDERS_DATA].as<std::vector<std::vector<double>>>();
35 }
```

Jun 08, 18 13:12 **FileReader.h** Page 1/1

```
1
2  #ifndef WORMS_FILEREADER_H
3  #define WORMS_FILEREADER_H
4
5  #include <fstream>
6  #include "MapObject.h"
7  #include <yaml.h>
8  #include <WeaponNames.h>
9  #include <ConfigFields.h>
10
11 // Clase que se encarga de manejar la carga de un mapa
12 class FileReader{
13 private:
14     std::fstream file;
15     std::string filename;
16 public:
17
18     explicit FileReader(const std::string &filename);
19
20     // Carga todos los componentes de un mapa desde un archivo YAML
21     void read(std::vector<std::vector<double>> &worms,
22               std::vector<std::vector<double>> &girders,
23               std::vector<int> &weps_ammo,
24               unsigned int &worm_life, std::string& background);
25 };
26
27
28 #endif //WORMS_FILEREADER_H
```

Jun 02, 18 18:24 **FileWriter.cpp** Page 1/1

```
1
2
3  #include "FileWriter.h"
4
5  FileWriter::FileWriter(const std::string &filename)
6          : file(filename, std::fstream::out | std::ios_base::trunc) {}
7
8
9  void FileWriter::save(std::vector<int> weapons,
10                        const std::vector<std::vector<double>> &worms,
11                        const std::vector<std::vector<double>> &girders,
12                        const unsigned int &worm_life, const std::string& backgrou
   nd) {
13     YAML::Emitter out;
14
15     out << YAML::BeginMap;
16
17     out << YAML::Key << BACKGROUND_IMAGE;
18     out << YAML::Value << background;
19
20     out << YAML::Key << WORMS_LIFE;
21     out << YAML::Value << worm_life;
22
23     out << YAML::Key << WEAPON_AMMO;
24
25     out << YAML::Value << YAML::BeginMap;
26
27     out << YAML::Key << BAZOOKA_NAME;
28     out << YAML::Value << weapons[0];
29     out << YAML::Key << MORTAR_NAME;
30     out << YAML::Value << weapons[1];
31     out << YAML::Key << GREEN_GRENADE_NAME;
32     out << YAML::Value << weapons[2];
33     out << YAML::Key << RED_GRENADE_NAME;
34     out << YAML::Value << weapons[3];
35     out << YAML::Key << BANANA_NAME;
36     out << YAML::Value << weapons[4];
37     out << YAML::Key << HOLY_GRENADE_NAME;
38     out << YAML::Value << weapons[9];
39     out << YAML::Key << DYNAMITE_NAME;
40     out << YAML::Value << weapons[8];
41     out << YAML::Key << BAT_NAME;
42     out << YAML::Value << weapons[6];
43     out << YAML::Key << AIR_ATTACK_NAME;
44     out << YAML::Value << weapons[5];
45     out << YAML::Key << TELEPORT_NAME;
46     out << YAML::Value << weapons[7];
47
48     out << YAML::EndMap;
49
50     out << YAML::Key << WORMS_DATA;
51     out << worms;
52
53     out << YAML::Key << GIRDERS_DATA;
54     out << girders;
55
56     out << YAML::EndMap;
57
58     file << out.c_str();
59 }
```

Jun 08, 18 13:12 **FileWriter.h** Page 1/1

```
1
2  #ifndef WORMS_FILEWRITER_H
3  #define WORMS_FILEWRITER_H
4
5  #include <fstream>
6  #include "MapObject.h"
7  #include <yaml.h>
8  #include <WeaponNames.h>
9  #include <ConfigFields.h>
10
11 // Clase que se encarga de manejar el guardado de un mapa
12 class FileWriter{
13 private:
14     std::fstream file;
15 public:
16     explicit FileWriter(const std::string &filename);
17
18     // Guarda todos los componentes de un mapa en un archivo YAML
19     void
20     save(std::vector<int> weapons,
21          const std::vector<std::vector<double>> &worms,
22          const std::vector<std::vector<double>> &girders,
23          const unsigned int &worm_life, const std::string& background);
24 };
25
26
27 #endif //WORMS_FILEWRITER_H
```

Jun 02, 18 18:24 **InvalidMapError.cpp** Page 1/1

```
1
2  #include <gtkmm/enums.h>
3  #include <gtkmm/messagedialog.h>
4  #include "InvalidMapError.h"
5
6  InvalidMapError::InvalidMapError(const char *message) noexcept : message(message
   ){
7  }
8
9  const char *InvalidMapError::what() const noexcept{
10     Gtk::Window dialog_window;
11     Gtk::MessageDialog dialog("Error al guardar archivo", false,Gtk::MESSAGE_WARNING);
12     dialog.set_transient_for(dialog_window);
13     dialog.set_secondary_text(message);
14     dialog.run();
15     return message;
16 }
17
18 InvalidMapError::~InvalidMapError() {
19 }
```

Jun 08, 18 13:12 **InvalidMapError.h** Page 1/1

```
1
2  #ifndef WORMS_INVALIDMAP_H
3  #define WORMS_INVALIDMAP_H
4
5
6  #include <exception>
7
8  // Clase que se encarga de lanzar una excepcion cuando el mapa a guardar es inva
   lido
9  class InvalidMapError : public std::exception{
10 private:
11     const char* message;
12 public:
13     InvalidMapError(const char *message) noexcept;
14
15     virtual const char *what() const noexcept;
16
17     ~InvalidMapError() override;
18 };
19
20
21 #endif //WORMS_INVALIDMAP_H
```

Jun 05, 18 14:07 **Map.cpp** Page 1/1

```
1
2  #include <yaml.h>
3  #include "Map.h"
4
5  void Map::erase(const int &index) {
6      if (!contained_objects.empty())
7          this->contained_objects.erase(contained_objects.begin() + index);
8  }
9
10 void Map::clean() {
11     this->contained_objects.clear();
12 }
13
14 void
15 Map::add(const unsigned int &id, const double &x, const double &y, const int &an
   gle) {
16     MapObject new_object(x, y, angle);
17     contained_objects.emplace_back(std::make_pair(id, new_object));
18 }
19
20 void Map::move(const int &index, const double &x,const double &y) {
21     MapObject &object = contained_objects[index].second;
22     object.updatePosition(x, y);
23 }
24
25 const int Map::turn(const unsigned int &index, unsigned int &id, const int &rota
   tion) {
26     MapObject &object = contained_objects[index].second;
27     id = contained_objects[index].first;
28     return object.turn(rotation);
29 }
30
31 const bool Map::isGirder(int &index) const {
32     return (contained_objects[index].first > 1);
33 }
34
35 void Map::getObjects(std::vector<std::vector<double>> &worms,
36                      std::vector<std::vector<double>> &girders) const {
37     for (auto &object : contained_objects) {
38         float x, y;
39         object.second.getPosition(x, y);
40         if (object.first == 1) {
41             std::vector<double> position;
42             position.push_back(x);
43             position.push_back(y);
44             worms.push_back(position);
45         } else {
46             std::vector<double> data;
47             data.push_back(object.first);
48             data.push_back(x);
49             data.push_back(y);
50             data.push_back(object.second.getAngle());
51             girders.push_back(data);
52         }
53     }
54 }
55
56 const int Map::getItemID(const int &index) const{
57     return contained_objects[index].first ;
58 }
59
60
61
```

Jun 08, 18 13:12 **Map.h** Page 1/1

```
1
2  #ifndef WORMS_MAPMODEL_H
3  #define WORMS_MAPMODEL_H
4
5
6  #include <utility>
7  #include <vector>
8  #include "MapObject.h"
9
10 // Clase que se encarga de modelar el mapa
11 class Map {
12     std::vector<std::pair<int, MapObject>> contained_objects;
13
14 public:
15     // Borra el objeto que se encuentra en la posicion index del vector
16     void erase(const int &index);
17
18     // Borra todos los objetos contenidos en el mapa
19     void clean();
20
21     // Agregar un objeto en la posicion (x,y)
22     void add(const unsigned int &id, const double &x, const double &y,
23              const int &angle = 0);
24
25     // Obtiene todos los objetos contenidos en el mapa separados por tipo
26     void getObjects(std::vector<std::vector<double>> &worms,
27                     std::vector<std::vector<double>> &girders) const;
28
29     // Mueve el objeto en la posicion index del vector hacia la posicion
30     // (x,y) del mapa
31     void move(const int &index, const double &x, const double &y);
32
33     // Devuelve verdadero si el objeto en la posicion index es una viga
34     const bool isGirder(int &index) const;
35
36     // Obtiene el tipo del objeto en la posicion index del vector
37     const int getItemID(const int &index) const;
38
39     // Gira el objeto en la posicion index del vector en un angulo indicado
40     const int
41     turn(const unsigned int &index, unsigned int &id, const int &rotation);
42 };
43
44
45 #endif //WORMS_MAPMODEL_H
```

Jun 02, 18 18:24 **MapObject.cpp** Page 1/1

```
1
2  #include <cstdlib>
3  #include "MapObject.h"
4
5  MapObject::MapObject(const float &x, const float &y, const int &angle) :
6          position(x,y), angle(angle) {}
7
8  void MapObject::updatePosition(const float &x, const float &y) {
9      position= Position(x,y);
10 }
11
12 int MapObject::turn(const int &rotation){
13     if (angle == 0)
14         angle = 180;
15     return angle = abs((angle+rotation)%180);
16 }
17
18 void MapObject::getPosition(float &x, float &y) const {
19     y=position.getY();
20     x=position.getX();
21 }
22
23 const int MapObject::getAngle() const {
24     return angle;
25 }
26
27
```

Jun 08, 18 13:12 **MapObject.h** Page 1/1

```
1
2  #ifndef WORMS_OBJECTMODEL_H
3  #define WORMS_OBJECTMODEL_H
4
5  #include <Position.h>
6
7  // Clase que modela un objeto contenido en el mapa
8  class MapObject {
9      Position position;
10     int angle;
11 public:
12     MapObject(const float &x, const float &y, const int &angle = 0);
13
14     // Actualiza la posicion en la que se encuentra el objeto
15     void updatePosition(const float &x, const float &y);
16
17     // Obtiene la posicion en la que se encuentra el objeto
18     void getPosition(float &x, float &y) const;
19
20     // Actualiza el angulo en la que se encuentra el objeto
21     const int getAngle() const;
22
23     // Gira el objeto la cantidad especificada
24     int turn(const int &rotation);
25 };
26
27
28 #endif //WORMS_OBJECTMODEL_H
```

Jun 02, 18 18:24 **Weapon.cpp** Page 1/1

```
1
2  #include "Weapon.h"
3
4  Weapon::Weapon(const int &default_ammo)
5          : default_ammo(default_ammo),
6            actual_ammo(default_ammo) {}
7
8  void Weapon::resetAmmo() {
9      actual_ammo = default_ammo;
10 }
11
12 void Weapon::setAmmo(const int &new_ammo) {
13     this->actual_ammo = new_ammo;
14 }
15
16 int Weapon::getAmmo() const {
17     return actual_ammo;
18 }
```

Jun 08, 18 13:12 **Weapon.h** Page 1/1

```
1
2  #ifndef WORMS_WEAPONMODEL_H
3  #define WORMS_WEAPONMODEL_H
4
5  // Clase que modela un arma
6  class Weapon {
7  private:
8      const int default_ammo;
9      int actual_ammo;
10 public:
11     explicit Weapon(const int &default_ammo);
12
13     // Establece el valor de la municion por defecto en el modelo
14     void resetAmmo();
15
16     // Establece el valor de la municion indicado en el modelo
17     void setAmmo(const int &new_ammo);
18
19     // Obtiene el valor actual de la municion
20     int getAmmo() const;
21 };
22
23
24 #endif //WORMS_WEAPONMODEL_H
```

Jun 03, 18 12:56 **FileBoxView.cpp** Page 1/1

```
1
2  #include "FileBoxView.h"
3
4  FileBoxView::FileBoxView(BaseObjectType *cobject,
5                           const Glib::RefPtr<Gtk::Builder> &builder)
6          : Gtk::Grid(cobject) {
7      builder->get_widget("btn_save", save);
8      builder->get_widget("btn_load", load);
9      builder->get_widget("btn_clean", new_map);
10 }
11
12 void FileBoxView::bindController(std::shared_ptr<FileBoxController> controller)
   {
13     this->file_box_controller = std::move(controller);
14
15     save->signal_clicked().connect(
16             sigc::mem_fun(*file_box_controller,
17                           &FileBoxController::onSaveClicked));
18
19     load->signal_clicked().connect(
20             sigc::mem_fun(*file_box_controller,
21                           &FileBoxController::onLoadClicked));
22
23     new_map->signal_clicked().connect(
24             sigc::mem_fun(*file_box_controller,
25                           &FileBoxController::onNewClicked));
26 }
```

Jun 08, 18 13:12 **FileBoxView.h** Page 1/1

```
1
2  #ifndef WORMS_FILEBOXVIEW_H
3  #define WORMS_FILEBOXVIEW_H
4
5  #include <gtkmm/builder.h>
6  #include <gtkmm/hvbox.h>
7  #include <gtkmm/button.h>
8  #include <gtkmm/grid.h>
9  #include "FileBoxController.h"
10
11 class FileBoxController;
12
13 // Clase que se encarga de manipular la zona de archivos
14 class FileBoxView : public Gtk::Grid {
15 private:
16     Gtk::Button *save;
17     Gtk::Button *load;
18     Gtk::Button *new_map;
19     std::shared_ptr<FileBoxController> file_box_controller;
20 public:
21     FileBoxView(BaseObjectType *cobject,
22                 const Glib::RefPtr<Gtk::Builder> &builder);
23
24     // Enlaza el controlador a la vista
25     void bindController(std::shared_ptr<FileBoxController> controller);
26 };
27
28
29 #endif //WORMS_FILEBOXVIEW_H
```

Jun 02, 18 13:44 **LifeView.cpp** Page 1/1

```
1
2  #include "LifeView.h"
3
4  LifeView::LifeView(BaseObjectType *cobject,
5                     const Glib::RefPtr<Gtk::Builder> &builder)
6          : Gtk::SpinButton(cobject),
7            default_hp(this->get_value()) {
8  }
9
10 void LifeView::reset() {
11     this->set_value(default_hp);
12 }
13
14 void LifeView::update(const unsigned int &new_life) {
15     this->set_value(new_life);
16 }
```

Jun 08, 18 13:12 **LifeView.h** Page 1/1

```
1
2  #ifndef WORMS_LIFEVIEW_H
3  #define WORMS_LIFEVIEW_H
4
5
6  #include <gtkmm/spinbutton.h>
7  #include <gtkmm/builder.h>
8
9  // Clase que se encarga de manipular la vista de la vida
10 class LifeView : public Gtk::SpinButton {
11 private:
12     const unsigned int default_hp;
13 public:
14     LifeView(BaseObjectType *cobject,
15              const Glib::RefPtr<Gtk::Builder> &builder);
16
17     // Establece el valor por defecto de la vida
18     void reset();
19
20     // Establece un nuevo valor a mostrar en la vista de la vida
21     void update(const unsigned int &new_life);
22 };
23
24
25 #endif //WORMS_LIFEVIEW_H
```

Jun 09, 18 16:29 **MapView.cpp** Page 1/3

```
1
2  #include <Path.h>
3  #include <gtkmm/adjustment.h>
4  #include <gtkmm/scrolledwindow.h>
5  #include <glibmm/main.h>
6  #include "MapView.h"
7  #include "GirderSize.h"
8
9  const std::string DEFAULT_BACKGROUND("default_background.png");
10
11 MapView::MapView(BaseObjectType *cobject,
12                  const Glib::RefPtr<Gtk::Builder> &builder)
13         : Gtk::Layout(cobject),
14           scroll_handler(*(Gtk::ScrolledWindow*)this->get_parent()){
15
16     add_events(Gdk::BUTTON_PRESS_MASK);
17     signal_button_press_event().connect(
18             sigc::mem_fun(*this, &MapView::onButtonClicked));
19
20     setInitialPosition();
21     changeBackground(BACKGROUND_PATH + DEFAULT_BACKGROUND);
22     initializeWormsImages();
23     initializeGirderImages();
24 }
25
26 bool MapView::onButtonClicked(GdkEventButton *button_event) {
27     controller->mapClickedSignal(button_event);
28     return true;
29 }
30
31 void MapView::setInitialPosition() {
32     guint width, height;
33     get_size(width, height);
34     ((Gtk::ScrolledWindow*) get_parent())->get_hadjustment()->set_value(width /
   2);
35     ((Gtk::ScrolledWindow*) get_parent())->get_vadjustment()->set_value(height);
36 }
37
38 void MapView::initializeGirderImages(){
39     std::vector<std::string> girder_3_imgs;
40     std::vector<std::string> girder_6_imgs;
41
42     for (int i = 0; i < 180; i = i + 10) {
43         girder_3_imgs.emplace_back(
44                 GIRDER_PATH + "3_" + std::to_string(i) + ".png");
45         girder_6_imgs.push_back(
46                 GIRDER_PATH + "6_" + std::to_string(i) + ".png");
47     }
48     objects_pallete.push_back(girder_3_imgs);
49     objects_pallete.push_back(girder_6_imgs);
50 }
51
52 void MapView::initializeWormsImages()  {
53     std::vector<std::string> worms_imgs;
54     worms_imgs.emplace_back(IMAGES_PATH + "/right_worm.png");
55     objects_pallete.push_back(worms_imgs);
56 }
57
58 void MapView::add(const unsigned int &id, const double &x, const double &y,
59                   const int &angle) {
60     Gtk::Image new_image(objects_pallete[id - id / 2 - 1][0]);
61     const Glib::RefPtr<Gdk::Pixbuf> &img = new_image.get_pixbuf();
62     int width = img->get_width();
63     int height = img->get_height();
64     double x_bound = x - width / 2;
65     double y_bound = y - height / 2;
```

```
66
67      put(new_image, x_bound, y_bound);
68      new_image.show();
69      contained_objects.push_back(std::move(new_image));
70      if (angle > 0){
71          sigc::slot<bool> my_slot = sigc::bind(sigc::mem_fun(*this, &MapView::tur
    n), id, angle, contained_objects.size()-1);
72          Glib::signal_idle().connect(my_slot);
73      }
74  }
75
76  void MapView::move(const int &index, const double &x, const double &y) {
77      if (!contained_objects.empty()) {
78          Gtk::Image &actual_object = contained_objects[index];
79          Gtk::Layout::move(actual_object, x - actual_object.get_width() / 2,
80                            y - actual_object.get_height() / 2);
81          actual_object.show();
82      }
83  }
84
85  bool MapView::turn(const unsigned int &id, const int &angle, const int &index) {
86      if (!contained_objects.empty()) {
87          Gtk::Image &image = contained_objects[index];
88          float x = child_property_x(image) + image.get_width() / 2;
89          float y = child_property_y(image) + image.get_height() / 2;
90          image.set(objects_pallete[id - id / 2 - 1][angle / 10]);
91
92          int height = GirderSize::getGirderHeightPixels(id, angle);
93          int width = GirderSize::getGirderWidthPixels(id, angle);
94          Gtk::Layout::move(image, x - width / 2, y - height / 2);
95      }
96      return false;
97  }
98
99  void MapView::erase(const int &index) {
100     if (!contained_objects.empty()) {
101         contained_objects[index].hide();
102         contained_objects.erase(contained_objects.begin() + index);
103     }
104 }
105
106 void MapView::clean() {
107     contained_objects.clear();
108 }
109
110 void MapView::bindController(MapController *map_controller) {
111     this->controller = map_controller;
112 }
113
114 void MapView::changeBackground(const std::string &path) {
115     background.clear();
116     Gtk::Image bg(path);
117     int img_width = bg.get_pixbuf()->get_width();
118     int img_height = bg.get_pixbuf()->get_height();
119     guint window_width, window_height;
120     this->get_size(window_width, window_height);
121     for (size_t x = 0; x < window_width; x += img_width) {
122         for (size_t y = 0; y < window_height; y += img_height) {
123             Gtk::Image image(path);
124             image.show();
125             put(image, x, y);
126             background.push_back(std::move(image));
127         }
128     }
129     redrawMap();
130 }
```



```
131
132 void MapView::redrawMap() {
133     for(Gtk::Image &object : contained_objects){
134         const Gtk::Allocation &alloc = object.get_allocation();
135         remove(object);
136         put(object,alloc.get_x(),alloc.get_y());
137     }
138     this->water.show(*this);
139 }
140
141 int MapView::select(const double &x, const double &y) {
142     Gdk::Rectangle new_object(x, y, 1, 1);
143     for (ssize_t i = contained_objects.size() - 1; i >= 0; i--) {
144         bool collision = contained_objects[i].intersect(new_object);
145         if (collision) {
146             return i;
147         }
148     }
149     return -1;
150 }
151
152 Glib::RefPtr<const Gdk::Pixbuf> MapView::getBackground() const{
153     return this->background[0].get_pixbuf();
154 }
155
156 void MapView::loadBackground(const std::string &name) {
157     changeBackground(BACKGROUND_PATH + name);
158 }
159
```

```
1
2   #ifndef WORMS_MAP_H
3   #define WORMS_MAP_H
4
5   #include <gtkmm/builder.h>
6   #include <gtkmm/layout.h>
7   #include <gtkmm/image.h>
8   #include "MapController.h"
9   #include "Water.h"
10  #include "ScrollHandler.h"
11
12  class MapController;
13
14  // Clase que se encarga de manipular la vista del mapa
15  class MapView : public Gtk::Layout {
16  private:
17      std::vector<Gtk::Image> contained_objects;
18      std::vector<std::vector<std::string>> objects_pallete;
19      MapController *controller;
20      std::vector<Gtk::Image> background;
21      Water water;
22      ScrollHandler scroll_handler;
23
24
25      // Inicializa el vector de imagenes de los worms
26      void initializeWormsImages();
27
28      // Inicializa el vector de imagenes de las vigas
29      void initializeGirderImages();
30
31      // Establece la posicion actual del mapa a mostrar
32      void setInitialPosition();
33
34      // Dibuja nuevamente el contenido del mapa
35      void redrawMap();
36
37  public:
38      MapView(BaseObjectType *cobject, const Glib::RefPtr<Gtk::Builder> &builder);
39
40      // Se ejecuta al clickear el mapa
41      bool onButtonClicked(GdkEventButton *button_event);
42
43      // Borra el objeto en la posiciÃ³n indicada
44      void erase(const int &index);
45
46      // Elimina todo el contenido del mapa
47      void clean();
48
49      // Enlaza el controlador a la vista
50      void bindController(MapController *map_controller);
51
52      // Agregar un nuevo objeto al mapa, en la posicion (x,y)
53      void add(const unsigned int &id, const double &x, const double &y,
54               const int &angle = 0);
55
56      // Gira el objeto seleccionado
57      bool turn(const unsigned int &id, const int &angle, const int &index);
58
59      // Cambia el fondo actual
60      void changeBackground(const std::string &path);
61
62      // Selecciona el objeto en la posiciÃ³n (x,y)
63      int select(const double &x, const double &y);
64
65      // Mueve el objeto seleccionado a la posicÃ³n (x.y)
66      void move(const int& index, const double &x, const double &y);
```



```
67
68      // Obtiene el nombre del fondo actual
69      Glib::RefPtr<const Gdk::Pixbuf> getBackground() const;
70
71      // Establece el fondo especificado por su nombre
72      void loadBackground(const std::string &name);
73
74  };
75
76  #endif //WORMS_MAP_H
```

Jun 03, 18 12:56 **ToolBoxView.cpp** Page 1/3

```
#include <gtkmm/builder.h>
#include <Path.h>
#include "ToolBoxView.h"

ToolBoxView::ToolBoxView(BaseObjectType *cobject,
                         const Glib::RefPtr<Gtk::Builder> &builder)
        : Gtk::Grid(cobject) {
    processing=false;

    builder->get_widget("tbtn_worm", worm);
    worm->set_active(true);
    builder->get_widget("tbtn_grd", girder_3m);
    builder->get_widget("tbtn_grd6", girder_6m);

    builder->get_widget("btn_move", move);
    builder->get_widget("btn_undo", erase);
    builder->get_widget("btn_turn_ccw", turnccw);
    builder->get_widget("btn_turn_cw", turncw);
    builder->get_widget("btn_bg", change_bg);
    builder->get_widget("btn_mode", mode);
    builder->get_widget("img_selected",selected);

    worm->signal_clicked().connect(sigc::bind<int>
            (sigc::mem_fun(*this, &ToolBoxView::onNewObjectClicked),
             WORM_BUTTON_ID));
    girder_3m->signal_clicked().connect(sigc::bind<int>
            (sigc::mem_fun(*this, &ToolBoxView::onNewObjectClicked),
             GIRDER_3_BUTTON_ID));

    girder_6m->signal_clicked().connect(sigc::bind<int>
            (sigc::mem_fun(*this, &ToolBoxView::onNewObjectClicked),
             GIRDER_6_BUTTON_ID));
}

void ToolBoxView::bindController(MapController *controller) {
    this->map_controller = controller;

    erase->signal_clicked().connect(
            sigc::mem_fun(*map_controller, &MapController::eraseSignal));

    move->signal_clicked().connect(
            sigc::mem_fun(*map_controller, &MapController::moveSignal));

    turnccw->signal_clicked().connect(
            sigc::mem_fun(*map_controller, &MapController::turnCCWSignal));

    turncw->signal_clicked().connect(
            sigc::mem_fun(*map_controller, &MapController::turnCWSignal));

    change_bg->signal_clicked().connect(
            sigc::mem_fun(*map_controller,
                          &MapController::changeBackgroundSignal));

    mode->signal_toggled().connect(
            sigc::mem_fun(*this, &ToolBoxView::changeMode));
}

void ToolBoxView::onNewObjectClicked(unsigned id) {
    if (!processing) {
        processing=true;
        if (id == WORM_BUTTON_ID) {
            if (worm->get_active()) {
                girder_3m->set_active(false);
                girder_6m->set_active(false);
            }
        } else if (id == GIRDER_3_BUTTON_ID) {
            if (girder_3m->get_active()) {
                worm->set_active(false);
                girder_6m->set_active(false);
            }
        } else {
            girder_3m->set_active(false);
            worm->set_active(false);
        }
        disableMovingItems();
        mode->set_active(false);
        map_controller->addModeSignal(id);
        leaveConsistent();
        processing=false;
    }
}

void ToolBoxView::enableMovingItems() {
    turncw->set_sensitive(true);
    turnccw->set_sensitive(true);
    move->set_sensitive(true);
    erase->set_sensitive(true);
}

void ToolBoxView::disableMovingItems() {
    turncw->set_sensitive(false);
    turnccw->set_sensitive(false);
    move->set_sensitive(false);
    erase->set_sensitive(false);
}

void ToolBoxView::changeMode() {
    worm->set_sensitive(!mode->get_active());
    girder_3m->set_sensitive(!mode->get_active());
    girder_6m->set_sensitive(!mode->get_active());
    if(!mode->get_active()){
        disableMovingItems();
    }
    map_controller->changeModeSignal();

}

void ToolBoxView::leaveConsistent() {
    if (!worm->get_active() && !girder_6m->get_active() && !girder_3m->get_activ
e()){
        processing=true;
        worm->set_active(true);
        map_controller->addModeSignal(WORM_BUTTON_ID);
    }
}

void ToolBoxView::showSelected(int id) {
    switch (id){
        case WORM_BUTTON_ID:
            selected->set(IMAGES_PATH+"/right_worm.png");
            selected->show();
            break;
        case GIRDER_3_BUTTON_ID:
            selected->set(IMAGES_PATH+"Girder/girder_3_selected.png");
            selected->show();
            break;
        case GIRDER_6_BUTTON_ID:
            selected->set(IMAGES_PATH+"Girder/girder_6_selected.png");
            selected->show();
            break;
        default:
```

Jun 03, 18 12:56 **ToolBoxView.cpp** Page 2/3

Jun 03, 18 12:56 **ToolBoxView.cpp** Page 3/3

```
132             hideSelected();
133             break;
134     }
135 }
136
137 void ToolBoxView::hideSelected() {
138     selected->hide();
139 }
140
141 void ToolBoxView::closeSelectionMode() {
142     disableMovingItems();
143     hideSelected();
144     mode->set_active(false);
145 }
146
```

Jun 08, 18 13:12 **ToolBoxView.h** Page 1/2

```
1
2  #ifndef WORMS_TOOLBOX_H
3  #define WORMS_TOOLBOX_H
4
5  #include <gtkmm/grid.h>
6  #include <gtkmm/button.h>
7  #include <gtkmm/layout.h>
8  #include <gtkmm/togglebutton.h>
9  #include <gtkmm/switch.h>
10 #include <gtkmm/hvbox.h>
11 #include "MapView.h"
12 #include "MapController.h"
13
14 #define WORM_BUTTON_ID 1
15 #define GIRDER_3_BUTTON_ID 3
16 #define GIRDER_6_BUTTON_ID 6
17 class MapController;
18
19 // Clase que contiene la vista de la botonera
20 class ToolBoxView : public Gtk::Grid {
21 private:
22     Gtk::Button *erase;
23     MapController *map_controller;
24     Gtk::ToggleButton *worm;
25     Gtk::ToggleButton *girder_3m;
26     Gtk::ToggleButton *girder_6m;
27     Gtk::Button *move;
28
29     Gtk::Button *turnccw;
30     Gtk::Button *turncw;
31     Gtk::Button *change_bg;
32     Gtk::ToggleButton *mode;
33     Gtk::Image* selected;
34     bool processing;
35
36     // Deja en un estado consistente la zona "Agregar"
37     void leaveConsistent();
38
39 public:
40     ToolBoxView(BaseObjectType *cobject,
41                 const Glib::RefPtr<Gtk::Builder> &builder);
42
43     // Se ejecuta cuando se selecciona un elemento de la zona "Agregar"
44     void onNewObjectClicked(unsigned int id);
45
46     // Habilita para el usuario la interacciÃ³n con las acciones de la zona
47     // "Seleccion"
48     void enableMovingItems();
49
50     // Deshabilita para el usuario la interacciÃ³n con las acciones de la zona
51     // "Seleccion"
52     void disableMovingItems();
53
54     // Enlaza la vista con el controlador
55     void bindController(MapController *controller);
56
57     // Alterna la vista entre el modo "Agregar" y modo "Seleccion"
58     void changeMode();
59
60     // Muestra el objeto seleccionado en el recuadro en la zona "Seleccion"
61     void showSelected(int id);
62
63     // VacÃ-a el recuadro en la zona "Seleccion"
64     void hideSelected();
65
66     // Sale del modo "Seleccion"
```

Jun 08, 18 13:12 **ToolBoxView.h** Page 2/2

```
67      void closeSelectionMode();
68  };
69
70
71  #endif //WORMS_TOOLBOX_H
```

Jun 03, 18 12:56 **WeaponView.cpp** Page 1/1

```
1   #include "WeaponView.h"
2
3   WeaponView::WeaponView(const Glib::RefPtr<Gtk::Builder> &builder,
4                          const unsigned int &id) {
5       builder->get_widget("sc_wep" + std::to_string(id), ammo_selector);
6       builder->get_widget("cb_wep" + std::to_string(id), infinite);
7
8       default_checkbox_state = infinite->get_active();
9       default_ammo_selector_value = ammo_selector->get_value();
10
11      ammo_selector->set_sensitive(!default_checkbox_state);
12
13      ammo_selector->signal_value_changed().connect(
14              sigc::mem_fun(*this, &WeaponView::onAmmoValueChanged));
15
16      infinite->signal_clicked().connect(
17              sigc::mem_fun(*this, &WeaponView::onCheckboxClicked));
18  }
19
20  void WeaponView::onAmmoValueChanged() {
21      controller->updateAmmo(ammo_selector->get_value());
22  }
23
24  void WeaponView::onCheckboxClicked() {
25      ammo_selector->set_sensitive(!infinite->get_active());
26      if (infinite->get_active()) {
27          controller->updateAmmo(-1);
28      } else
29          controller->updateAmmo(ammo_selector->get_value());
30  }
31
32  void WeaponView::resetAmmo() {
33      ammo_selector->set_sensitive(!default_checkbox_state);
34      ammo_selector->set_value(default_ammo_selector_value);
35      infinite->set_active(default_checkbox_state);
36  }
37
38  void WeaponView::bindController(WeaponController *controller) {
39      this->controller = controller;
40  }
41
42  const int WeaponView::getInitialAmmo() {
43      return default_checkbox_state ? -1 : default_ammo_selector_value;
44  }
45
46  void WeaponView::setAmmo(const int &ammo) {
47      if (ammo < 0) {
48          infinite->set_active(true);
49          ammo_selector->set_sensitive(false);
50      } else {
51          infinite->set_active(false);
52          ammo_selector->set_sensitive(true);
53          ammo_selector->set_value(ammo);
54      }
55
56  }
```

Jun 08, 18 13:12 **WeaponView.h** Page 1/1

```
1
2  #ifndef WORMS_WEP_H
3  #define WORMS_WEP_H
4
5
6  #include <gtkmm/hvbox.h>
7  #include <gtkmm/scale.h>
8  #include <gtkmm/checkbutton.h>
9  #include <gtkmm/builder.h>
10 #include "WeaponController.h"
11 class WeaponController;
12
13 // Clase que contiene la vista de cada arma
14 class WeaponView {
15 private:
16     Gtk::Scale *ammo_selector;
17     Gtk::CheckButton *infinite;
18     bool default_checkbox_state;
19     int default_ammo_selector_value;
20     WeaponController *controller;
21
22 public:
23     WeaponView(const Glib::RefPtr<Gtk::Builder> &builder,
24                const unsigned int &id);
25
26     // Al cambiar el valor del scale se llama a este mÃ©todo.
27     void onAmmoValueChanged();
28
29     // Al cambiar el estado del checkbox  se llama a este mÃ©todo.
30     void onCheckboxClicked();
31
32     // Muestra la municiÃ³n predeterminada de esta arma
33     void resetAmmo();
34
35     // Enlaza la vista al controlador
36     void bindController(WeaponController *controller);
37
38     // Obtiene la municiÃ³n inicial
39     const int getInitialAmmo();
40
41     // Establece la municiÃ³n a mostrar
42     void setAmmo(const int &ammo);
43 };
44
45
46 #endif //WORMS_WEP_H
```

Jun 09, 18 18:50 **Table of Content** Page 1/1

```
1  Table of Contents
2    1 FileBoxController.cpp sheets   1 to   1 ( 1) pages   1-  2   93 lines
3    2 FileBoxController.h. sheets   2 to   2 ( 1) pages   3-  3   39 lines
4    3 MapController.cpp... sheets   2 to   3 ( 2) pages   4-  6  159 lines
5    4 MapController.h..... sheets   4 to   4 ( 1) pages   7-  7   59 lines
6    5 UsablesController.cpp sheets   4 to   5 ( 2) pages   8-  9   67 lines
7    6 UsablesController.h. sheets   5 to   5 ( 1) pages  10- 10   43 lines
8    7 WeaponController.cpp sheets   6 to   6 ( 1) pages  11- 11   24 lines
9    8 WeaponController.h.. sheets   6 to   6 ( 1) pages  12- 12   33 lines
10   9 main.cpp............ sheets   7 to   7 ( 1) pages  13- 13   39 lines
11  10 Editor.cpp.......... sheets   7 to   7 ( 1) pages  14- 14   22 lines
12  11 Editor.h............ sheets   8 to   8 ( 1) pages  15- 15   28 lines
13  12 FileReader.cpp...... sheets   8 to   8 ( 1) pages  16- 16   36 lines
14  13 FileReader.h........ sheets   9 to   9 ( 1) pages  17- 17   29 lines
15  14 FileWriter.cpp...... sheets   9 to   9 ( 1) pages  18- 18   60 lines
16  15 FileWriter.h........ sheets  10 to  10 ( 1) pages  19- 19   28 lines
17  16 InvalidMapError.cpp. sheets  10 to  10 ( 1) pages  20- 20   20 lines
18  17 InvalidMapError.h... sheets  11 to  11 ( 1) pages  21- 21   22 lines
19  18 Map.cpp............. sheets  11 to  11 ( 1) pages  22- 22   62 lines
20  19 Map.h............... sheets  12 to  12 ( 1) pages  23- 23   46 lines
21  20 MapObject.cpp....... sheets  12 to  12 ( 1) pages  24- 24   28 lines
22  21 MapObject.h......... sheets  13 to  13 ( 1) pages  25- 25   29 lines
23  22 Weapon.cpp.......... sheets  13 to  13 ( 1) pages  26- 26   19 lines
24  23 Weapon.h............ sheets  14 to  14 ( 1) pages  27- 27   25 lines
25  24 FileBoxView.cpp..... sheets  14 to  14 ( 1) pages  28- 28   27 lines
26  25 FileBoxView.h....... sheets  15 to  15 ( 1) pages  29- 29   30 lines
27  26 LifeView.cpp........ sheets  15 to  15 ( 1) pages  30- 30   17 lines
28  27 LifeView.h.......... sheets  16 to  16 ( 1) pages  31- 31   26 lines
29  28 MapView.cpp......... sheets  16 to  17 ( 2) pages  32- 34  160 lines
30  29 MapView.h........... sheets  18 to  18 ( 1) pages  35- 36   77 lines
31  30 ToolBoxView.cpp..... sheets  19 to  20 ( 2) pages  37- 39  147 lines
32  31 ToolBoxView.h....... sheets  20 to  21 ( 2) pages  40- 41   72 lines
33  32 WeaponView.cpp...... sheets  21 to  21 ( 1) pages  42- 42   57 lines
34  33 WeaponView.h........ sheets  22 to  22 ( 1) pages  43- 43   47 lines
```