

jun 19, 18 14:51

## FileBoxController.cpp

Page 1/2

```

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

```

jun 19, 18 14:51

## FileBoxController.cpp

Page 2/2

```

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

```

jun 07, 18 23:33

## 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 #endif //WORMS_FILECONTROLLER_H

```

jun 19, 18 14:51

## MapController.cpp

Page 1/3

```

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

```

jun 19, 18 14:51

## MapController.cpp

Page 2/3

```

67 }
68
69 void MapController::turnCWSignal() {
70     turn(-10);
71 }
72
73 void MapController::mapClickedSignal(GdkEventButton *event_button) {
74     if (actual_mode == MOVE_CMD_ID) {
75         this->model.move(index_object_selected, event_button->x,
76             event_button->y);
77         this->view->move(index_object_selected, event_button->x,
78             event_button->y);
79         actual_mode = SELECT_MODE_ID;
80     } else if (actual_mode == SELECT_MODE_ID) {
81         this->index_object_selected = view->select(event_button->x,
82             event_button->y);
83         if (index_object_selected > -1) {
84             toolbox->enableMovingItems();
85             toolbox->showSelected(model.getItemID(index_object_selected));
86         } else {
87             toolbox->disableMovingItems();
88             toolbox->hideSelected();
89         }
90         //cambio de estado del toolbox llama a add mode
91         actual_mode = SELECT_MODE_ID;
92     } else {
93         this->model.add(item_id_to_add, event_button->x, event_button->y);
94         this->view->add(item_id_to_add, event_button->x, event_button->y);
95     }
96 }
97
98 void MapController::getObjects(std::vector<std::vector<double>> &worms,
99     std::vector<std::vector<double>> &girders) const {
100     model.getObjects(worms, girders);
101     if (worms.empty()) {
102         throw InvalidMapError("El mapa actual no contiene worms");
103     }
104     if (girders.empty()) {
105         throw InvalidMapError("El mapa actual no contiene vigas");
106     }
107
108     ViewPositionTransformer transformer(*view);
109     for (std::vector<double> &worm : worms) {
110         Position position(worm[0], worm[1]);
111         Position new_pos = transformer.transformToPosition(position);
112         worm[0] = new_pos.getX();
113         worm[1] = new_pos.getY();
114     }
115     for (std::vector<double> &girder : girders) {
116         Position position(girder[1], girder[2]);
117         Position new_pos = transformer.transformToPosition(position);
118         girder[1] = new_pos.getX();
119         girder[2] = new_pos.getY();
120     }
121 }
122
123 void MapController::loadObjects(std::vector<std::vector<double>> &worms,
124     std::vector<std::vector<double>> &girders) {
125     newMapSignal();
126     ViewPositionTransformer transformer(*view);
127     for (std::vector<double> &girder : girders) {
128         Position position(girder[1], girder[2]);
129         Position new_pos = transformer.transformToScreen(position);
130         girder[1] = new_pos.getX();
131         girder[2] = new_pos.getY();
132         this->model.add(girder[0], girder[1], girder[2], girder[3]);

```

jun 19, 18 14:51

## MapController.cpp

Page 3/3

```

133         this->view->add(girder[0], girder[1], girder[2], girder[3]);
134     }
135     for (std::vector<double> &worm : worms) {
136         Position position(worm[0], worm[1]);
137         Position new_pos = transformer.transformToScreen(position);
138         worm[0] = new_pos.getX();
139         worm[1] = new_pos.getY();
140         this->model.add(1, worm[0], worm[1]);
141         this->view->add(1, worm[0], worm[1]);
142     }
143     this->view->setInitialPosition();
144 }
145
146 void MapController::changeBackgroundSignal() const {
147     this->background_dialog->set_current_folder(BACKGROUND_PATH);
148     int result = this->background_dialog->run();
149     if (result == Gtk::RESPONSE_OK) {
150         std::string path = this->background_dialog->get_filename();
151         this->view->changeBackground(path);
152     }
153     this->background_dialog->hide();
154 }
155
156 Glib::RefPtr<const Gdk::Pixbuf> MapController::getBackground() const {
157     return view->getBackground();
158 }
159
160 void MapController::loadBackground(Glib::RefPtr<Gdk::Pixbuf> &background) {
161     view->loadBackground(background);
162 }

```

jun 19, 18 14:51

## MapController.h

Page 1/2

```

1
2 #ifndef WORMS_MAPCONTROLLER_H
3 #define WORMS_MAPCONTROLLER_H
4
5 #include <gtkmm/filechooserdialog.h>
6 #include <string>
7 #include <vector>
8 #include "MapView.h"
9 #include "Map.h"
10 #include "ToolBoxView.h"
11
12 class MapView;
13 class ToolBoxView;
14
15 // Clase que se encarga de comunicar la vista con el modelo, y a su vez, se
16 // comunica con el resto del programa
17 class MapController {
18     Map model;
19     MapView *view;
20     ToolBoxView *toolBox;
21     unsigned int item_id_to_add;
22     unsigned int actual_mode;
23     int index_object_selected;
24     Gtk::FileChooserDialog* background_dialog;
25
26     void turn(const int &rotation);
27
28 public:
29     /* Constructor */
30     MapController(Map model, const Glib::RefPtr<Gtk::Builder> &builder);
31
32     /* Cambia al modo de agregado, en donde el objeto
33     * a agregar es el de id */
34     void addModeSignal(const unsigned int &id);
35
36     /* Envia una señal de borrado */
37     void eraseSignal();
38
39     /* Envia una señal de nuevo mapa */
40     void newMapSignal();
41
42     /* Envia una señal de movimiento */
43     void moveSignal();
44
45     /* Envia una señal de rotacion anti horario */
46     void turnCCWSignal();
47
48     /* Envia una señal de click sobre el mapa */
49     void mapClickedSignal(GdkEventButton *event_button);
50
51     /* Obtiene los objetos del mapa */
52     void getObjects(std::vector<std::vector<double>> &worms,
53                    std::vector<std::vector<double>> &girders) const;
54
55     /* Carga los worms y las vigas en el mapa */
56     void loadObjects(std::vector<std::vector<double>> &worms,
57                    std::vector<std::vector<double>> &girders);
58
59     /* Envia una señal de rotacion horaria */
60     void turnCWSignal();
61
62     /* Envia una señal de cambio de imagen de fondo */
63     void changeBackgroundSignal() const;
64
65     /* Envia una señal de cambio de modo */
66     void changeModeSignal();

```

jun 19, 18 14:51

## MapController.h

Page 2/2

```

67
68     /* Devuelve la imagen de fondo */
69     Glib::RefPtr<const Gdk::Pixbuf> getBackground() const;
70
71     /* Carga la imagen de fondo */
72     void loadBackground(Glib::RefPtr<Gdk::Pixbuf> &background);
73 };
74
75
76 #endif //WORMS_MAPCONTROLLER_H

```

jun 10, 18 19:29

## UsablesController.cpp

Page 1/1

```

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

```

jun 10, 18 19:29

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

```

jun 04, 18 21:54

## WeaponController.cpp

Page 1/1

```

1
2 #include "WeaponController.h"
3
4 WeaponController::WeaponController(std::shared_ptr<WeaponView> View,
5                                   std::shared_ptr<Weapon> model)
6     : weapon_view(std::move(View)),
7       weapon_model(std::move(model)) {
8     weapon_view->bindController(this);
9 }
10
11 void WeaponController::resetAmmo() {
12     weapon_view->resetAmmo();
13     weapon_model->resetAmmo();
14 }
15
16 void WeaponController::updateAmmo(const int &ammo) {
17     weapon_model->setAmmo(ammo);
18     weapon_view->setAmmo(ammo);
19 }
20
21 int WeaponController::getAmmo() {
22     return weapon_model->getAmmo();
23 }

```

jun 07, 18 23:33

## WeaponController.h

Page 1/1

```

1
2 #ifndef WORMS_WEAPONCONTROLLER_H
3 #define WORMS_WEAPONCONTROLLER_H
4
5 #include "WeaponView.h"
6 #include "Weapon.h"
7 class WeaponView;
8
9 // Clase que se encarga de manejar la informacion del arma entre el modelo
10 // y la vista
11 class WeaponController {
12 private:
13     std::shared_ptr<WeaponView> weapon_view;
14     std::shared_ptr<Weapon> weapon_model;
15 public:
16     WeaponController(std::shared_ptr<WeaponView>,
17                     std::shared_ptr<Weapon>
18                     model);
19
20     // Indica a la vista y al modelo que deben resetear la municion
21     void resetAmmo();
22
23     // Indica a la vista y al modelo que deben establecer un nuevo valor de
24     // municion especificado
25     void updateAmmo(const int &ammo);
26
27     // Obtiene el valor de la municion desde el modelo
28     int getAmmo();
29 };
30
31
32 #endif //WORMS_WEAPONCONTROLLER_H

```

jun 10, 18 19:29

main.cpp

Page 1/1

```

1  #include <gtkmm/application.h>
2  #include <gtkmm/builder.h>
3  #include <giomm.h>
4  #include <iostream>
5  #include <gtkmm/scrolledwindow.h>
6  #include <gtkmm/window.h>
7  #include "Editor.h"
8  #include "Path.h"
9
10 int main() {
11     Glib::RefPtr<Gtk::Application> app = Gtk::Application::create();
12     Glib::RefPtr<Gtk::Builder> refBuilder = Gtk::Builder::create();
13     try {
14         refBuilder->add_from_file(GLADE_PATH + "editor.glade");
15     }
16     catch(const Glib::FileError &ex) {
17         std::cerr << "FileError: " << ex.what() << std::endl;
18         return 1;
19     }
20     catch(const Glib::MarkupError &ex) {
21         std::cerr << "MarkupError: " << ex.what() << std::endl;
22         return 1;
23     }
24     catch(const Gtk::BuilderError &ex) {
25         std::cerr << "BuilderError: " << ex.what() << std::endl;
26         return 1;
27     }
28
29     Editor *mainWindow = nullptr;
30     refBuilder->get_widget_derived("main_window", mainWindow);
31     if (mainWindow) {
32         mainWindow->set_title(EDITOR_WINDOW_NAME);
33         mainWindow->set_icon_from_file(ICON_PATH);
34         app->run(*mainWindow);
35         delete mainWindow;
36     }
37     return 0;
38 }

```

jun 10, 18 19:29

Editor.cpp

Page 1/1

```

1
2  #include "Editor.h"
3
4  typedef const Glib::RefPtr<Gtk::Builder> Builder;
5
6  Editor::Editor(BaseObjectType *cobject, Builder &builder):
7      Gtk::Window(cobject),
8      usables_controller(builder) {
9      maximize();
10     builder->get_widget("map_window", map_window);
11
12     std::shared_ptr<MapController> map_controller
13         (new MapController(map_model, builder));
14
15     builder->get_widget_derived("filebox", filebox);
16     std::shared_ptr<FileBoxController> filebox_controller(
17         new FileBoxController(usables_controller, map_controller, builder));
18     filebox->bind_controller(filebox_controller);
19
20     show_all_children();
21 }

```

jun 10, 18 19:29

## 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 #endif //WORMS_EDITOR_H

```

jun 21, 18 12:36

## FileReader.cpp

Page 1/1

```

1
2 #include "FileReader.h"
3 #include <map>
4 #include <string>
5 #include <vector>
6 #include <giomm/memoryinputstream.h>
7 #include "Buffer.h"
8
9 FileReader::FileReader(const std::string &filename):
10     file(filename, std::fstream::in),
11     filename(filename) {}
12
13 void FileReader::read(std::vector<std::vector<double>> &worms,
14                     std::vector<std::vector<double>> &girders,
15                     std::vector<int> &weps_amm,
16                     unsigned int &worms_life,
17                     Glib::RefPtr<Gdk::Pixbuf> &background) {
18     YAML::Node config = YAML::LoadFile(filename);
19
20     worms_life = config[WORMS_LIFE].as<unsigned int>();
21
22     std::map<std::string, int> ammo =
23         config[WEAPON_AMMO].as<std::map<std::string, int>>();
24
25     weps_amm.push_back(ammo[BAZOOKA_NAME]);
26     weps_amm.push_back(ammo[MORTAR_NAME]);
27     weps_amm.push_back(ammo[GREEN_GRENADE_NAME]);
28     weps_amm.push_back(ammo[RED_GRENADE_NAME]);
29     weps_amm.push_back(ammo[BANANA_NAME]);
30     weps_amm.push_back(ammo[AIR_ATTACK_NAME]);
31     weps_amm.push_back(ammo[BAT_NAME]);
32     weps_amm.push_back(ammo[TELEPORT_NAME]);
33     weps_amm.push_back(ammo[DYNAMITE_NAME]);
34     weps_amm.push_back(ammo[HOLY_GRENADE_NAME]);
35
36     worms = config[WORMS_DATA].as<std::vector<std::vector<double>>>();
37
38     girders = config[GIRDERS_DATA].as<std::vector<std::vector<double>>>();
39
40     std::vector<int> backgrounds =
41         config[BACKGROUND_IMAGE].as<std::vector<int>>();
42     Buffer buffer(backgrounds.size());
43     for (const int &actual : backgrounds) {
44         buffer.setNext(actual);
45     }
46
47     auto stream = Gio::MemoryInputStream::create();
48     stream->add_data(buffer.getPointer(), buffer.getMaxSize());
49     background = Gdk::Pixbuf::create_from_stream(stream);
50 }

```



jun 19, 18 14:51

## 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 #include <string>
11 #include <vector>
12 #include <gtkmm/image.h>
13
14 // Clase que se encarga de manejar la carga de un mapa
15 class FileReader{
16 private:
17     std::fstream file;
18     std::string filename;
19
20 public:
21     explicit FileReader(const std::string &filename);
22
23     // Carga todos los componentes de un mapa desde un archivo YAML
24     void read(std::vector<std::vector<double>> &worms,
25             std::vector<std::vector<double>> &girders,
26             std::vector<int> &weps_amm,
27             unsigned int &worm_life, Glib::RefPtr<Gdk::Pixbuf> &background);
28 };
29
30
31 #endif //WORMS_FILEREADER_H

```

jun 19, 18 14:51

## FileWriter.cpp

Page 1/2

```

1
2
3 #include "FileWriter.h"
4 #include <string>
5 #include <vector>
6
7 FileWriter::FileWriter(const std::string &filename):
8     file(filename, std::fstream::out | std::ios_base::trunc) {}
9
10
11 void FileWriter::save(std::vector<int> weapons,
12                     const std::vector<std::vector<double>> &worms,
13                     const std::vector<std::vector<double>> &girders,
14                     const unsigned int &worm_life,
15                     Glib::RefPtr<const Gdk::Pixbuf> &background) {
16     YAML::Emitter out;
17
18     out << YAML::BeginMap;
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::Key << BACKGROUND_IMAGE;
57     out << YAML::Value << YAML::Flow << YAML::BeginSeq;
58
59     char* image;
60     gsize size;
61     background->save_to_buffer(image, size);
62
63     for (size_t i = 0; i < size; i++){
64         out << (int)image[i];
65     }
66

```

jun 19, 18 14:51

## FileWriter.cpp

Page 2/2

```

67     out << YAML::EndSeq;
68     out << YAML::EndMap;
69
70     file << out.c_str();
71 }

```

jun 19, 18 14:51

## 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 #include <vector>
11 #include <string>
12 #include <gtkmm/image.h>
13
14 // Clase que se encarga de manejar el guardado de un mapa
15 class FileWriter{
16 private:
17     std::fstream file;
18
19 public:
20     explicit FileWriter(const std::string &filename);
21
22     // Guarda todos los componentes de un mapa en un archivo YAML
23     void
24     save(std::vector<int> weapons,
25         const std::vector<std::vector<double>> &worms,
26         const std::vector<std::vector<double>> &girders,
27         const unsigned int &worm_life, Glib::RefPtr<const Gdk::Pixbuf>& backgro
28     und);
29 };
30 #endif //WORMS_FILEWRITER_H

```

jun 10, 18 19:29

## 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:
7     message(message) {}
8
9 const char *InvalidMapError::what() const noexcept{
10     Gtk::Window dialog_window;
11     Gtk::MessageDialog dialog("Error al guardar archivo",
12                             false, Gtk::MESSAGE_WARNING);
13     dialog.set_transient_for(dialog_window);
14     dialog.set_secondary_text(message);
15     dialog.run();
16     return message;
17 }
18
19 InvalidMapError::~InvalidMapError() {
20 }

```

jun 10, 18 19:29

## 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
9 // cuando el mapa a guardar es invalido
10 class InvalidMapError : public std::exception{
11 private:
12     const char* message;
13
14 public:
15     explicit InvalidMapError(const char *message) noexcept;
16
17     virtual const char *what() const noexcept;
18
19     ~InvalidMapError();
20 };
21
22
23 #endif //WORMS_INVALIDMAP_H

```

jun 10, 18 19:29

## Map.cpp

Page 1/1

```

1
2 #include <vector>
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,
16         const double &y, const int &angle) {
17     MapObject new_object(x, y, angle);
18     contained_objects.emplace_back(std::make_pair(id, new_object));
19 }
20
21 void Map::move(const int &index, const double &x, const double &y) {
22     MapObject &object = contained_objects[index].second;
23     object.updatePosition(x, y);
24 }
25
26 const int Map::turn(const unsigned int &index,
27                    unsigned int &id, const int &rotation) {
28     MapObject &object = contained_objects[index].second;
29     id = contained_objects[index].first;
30     return object.turn(rotation);
31 }
32
33 const bool Map::isGirder(int &index) const {
34     return (contained_objects[index].first > 1);
35 }
36
37 void Map::getObjects(std::vector<std::vector<double>> &worms,
38                    std::vector<std::vector<double>> &girders) const {
39     for (auto &object : contained_objects) {
40         float x, y;
41         object.second.getPosition(x, y);
42         if (object.first == 1) {
43             std::vector<double> position;
44             position.push_back(x);
45             position.push_back(y);
46             worms.push_back(position);
47         } else {
48             std::vector<double> data;
49             data.push_back(object.first);
50             data.push_back(x);
51             data.push_back(y);
52             data.push_back(object.second.getAngle());
53             girders.push_back(data);
54         }
55     }
56 }
57
58 const int Map::getItemID(const int &index) const {
59     return contained_objects[index].first;
60 }

```

jun 07, 18 23:33

## 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 20:07

## 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 07, 18 23:33

## 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 20:07

## Weapon.cpp

Page 1/1

```

1
2 #include "Weapon.h"
3
4 Weapon::Weapon(const int &default_amm)
5     : default_amm(default_amm),
6     actual_amm(default_amm) {}
7
8 void Weapon::resetAmm() {
9     actual_amm = default_amm;
10 }
11
12 void Weapon::setAmm(const int &new_amm) {
13     this->actual_amm = new_amm;
14 }
15
16 int Weapon::getAmm() const {
17     return actual_amm;
18 }

```

jun 07, 18 23:33

## 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_amm;
9     int actual_amm;
10 public:
11     explicit Weapon(const int &default_amm);
12
13     // Establece el valor de la municion por defecto en el modelo
14     void resetAmm();
15
16     // Establece el valor de la municion indicado en el modelo
17     void setAmm(const int &new_amm);
18
19     // Obtiene el valor actual de la municion
20     int getAmm() const;
21 };
22
23
24 #endif //WORMS_WEAPONMODEL_H

```

jun 10, 18 19:29

## 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 07, 18 23:33

## 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 13:26

## 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 07, 18 23:33

## 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 19, 18 14:51

## MapView.cpp

Page 1/3

```

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

```

jun 19, 18 14:51

## MapView.cpp

Page 2/3

```

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(
72             *this, &MapView::turn), id, angle, contained_objects.size() - 1);
73         Glib::signal_idle().connect(my_slot);
74     }
75 }
76
77
78 void MapView::move(const int &index, const double &x, const double &y) {
79     if (!contained_objects.empty()) {
80         Gtk::Image &actual_object = contained_objects[index];
81         Gtk::Layout::move(actual_object, x - actual_object.get_width() / 2,
82             y - actual_object.get_height() / 2);
83         actual_object.show();
84     }
85 }
86
87 bool MapView::turn(const unsigned int &id, const int &angle, const int &index) {
88     if (!contained_objects.empty()) {
89         Gtk::Image &image = contained_objects[index];
90         float x = child_property_x(image) + image.get_width() / 2;
91         float y = child_property_y(image) + image.get_height() / 2;
92         image.set(objects_pallette[id - id / 2 - 1][angle / 10]);
93
94         int height = GirderSize::getGirderHeightPixels(id, angle);
95         int width = GirderSize::getGirderWidthPixels(id, angle);
96         Gtk::Layout::move(image, x - width / 2, y - height / 2);
97     }
98     return false;
99 }
100
101 void MapView::erase(const int &index) {
102     if (!contained_objects.empty()) {
103         contained_objects[index].hide();
104         contained_objects.erase(contained_objects.begin() + index);
105     }
106 }
107
108 void MapView::clean() {
109     contained_objects.clear();
110     this->setInitialPosition();
111 }
112
113 void MapView::bindController(MapController *map_controller) {
114     this->controller = map_controller;
115 }
116
117 void MapView::changeBackground(const std::string &path) {
118     Gtk::Image bg(path);
119     loadBackground(bg.get_pixbuf());
120 }
121
122 void MapView::redrawMap() {
123     for (Gtk::Image &object : contained_objects) {
124         const Gtk::Allocation &alloc = object.get_allocation();
125         remove(object);
126         put(object, alloc.get_x(), alloc.get_y());
127     }
128     this->water.show(*this);
129 }
130
131 int MapView::select(const double &x, const double &y) {
132     Gdk::Rectangle new_object(x, y, 1, 1);

```

jun 19, 18 14:51

## MapView.cpp

Page 3/3

```

133     for (ssize_t i = contained_objects.size() - 1; i >= 0; i--) {
134         bool collision = contained_objects[i].intersect(new_object);
135         if (collision) {
136             return i;
137         }
138     }
139     return -1;
140 }
141
142 Glib::RefPtr<const Gdk::Pixbuf> MapView::getBackground() const{
143     return this->background[0].get_pixbuf();
144 }
145
146 void MapView::loadBackground(Glib::RefPtr<Gdk::Pixbuf> pixbuf) {
147     background.clear();
148     int img_width = pixbuf->get_width();
149     int img_height = pixbuf->get_height();
150     guint window_width, window_height;
151     this->get_size(window_width, window_height);
152     for (size_t x = 0; x < window_width; x += img_width) {
153         for (size_t y = 0; y < window_height; y += img_height) {
154             Gtk::Image image(pixbuf);
155             image.show();
156             put(image, x, y);
157             background.push_back(std::move(image));
158         }
159     }
160     redrawMap();
161 }
162

```

jun 19, 18 14:51

## MapView.h

Page 1/2

```

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

```

jun 19, 18 14:51

## MapView.h

Page 2/2

```

67 // Mueve el objeto seleccionado a la posición (x,y)
68 void move(const int& index, const double &x, const double &y);
69
70 // Obtiene el nombre del fondo actual
71 Glib::RefPtr<const Gdk::Pixbuf> getBackground() const;
72
73 // Establece el fondo especificado
74 void loadBackground(Glib::RefPtr<Gdk::Pixbuf> pixbuf);
75 };
76
77 #endif //WORMS_MAP_H

```

jun 12, 18 20:48

## ToolBoxView.cpp

Page 1/3

```

1
2 #include <gtkmm/builder.h>
3 #include <Path.h>
4 #include "ToolBoxView.h"
5
6 ToolBoxView::ToolBoxView(BaseObjectType *cobject,
7                           const Glib::RefPtr<Gtk::Builder> &builder)
8     : Gtk::Grid(cobject) {
9     processing=false;
10
11     builder->get_widget("btn_worm", worm);
12     worm->set_active(true);
13     builder->get_widget("btn_grd", girder_3m);
14     builder->get_widget("btn_grd6", girder_6m);
15
16     builder->get_widget("btn_move", move);
17     builder->get_widget("btn_undo", erase);
18     builder->get_widget("btn_turn_ccw", turnccw);
19     builder->get_widget("btn_turn_cw", turncw);
20     builder->get_widget("btn_bg", change_bg);
21     builder->get_widget("btn_mode", mode);
22     builder->get_widget("img_selected", selected);
23
24     worm->signal_clicked().connect(sigc::bind<int>
25                                   (sigc::mem_fun(*this, &ToolBoxView::onNewObjectClicked),
26                                   WORM_BUTTON_ID));
27     girder_3m->signal_clicked().connect(sigc::bind<int>
28                                       (sigc::mem_fun(*this, &ToolBoxView::onNewObjectClicked),
29                                       GIRDER_3_BUTTON_ID));
30
31     girder_6m->signal_clicked().connect(sigc::bind<int>
32                                       (sigc::mem_fun(*this, &ToolBoxView::onNewObjectClicked),
33                                       GIRDER_6_BUTTON_ID));
34 }
35
36 void ToolBoxView::bindController(MapController *controller) {
37     this->map_controller = controller;
38
39     erase->signal_clicked().connect(
40         sigc::mem_fun(*map_controller, &MapController::eraseSignal));
41
42     move->signal_clicked().connect(
43         sigc::mem_fun(*map_controller, &MapController::moveSignal));
44
45     turnccw->signal_clicked().connect(
46         sigc::mem_fun(*map_controller, &MapController::turnCCWSignal));
47
48     turncw->signal_clicked().connect(
49         sigc::mem_fun(*map_controller, &MapController::turnCWSignal));
50
51     change_bg->signal_clicked().connect(
52         sigc::mem_fun(*map_controller,
53                       &MapController::changeBackgroundSignal));
54
55     mode->signal_toggled().connect(
56         sigc::mem_fun(*this, &ToolBoxView::changeMode));
57 }
58
59 void ToolBoxView::onNewObjectClicked(unsigned id) {
60     if (!processing) {
61         processing=true;
62         if (id == WORM_BUTTON_ID) {
63             if (worm->get_active()) {
64                 girder_3m->set_active(false);
65                 girder_6m->set_active(false);
66             }

```

jun 12, 18 20:48

## ToolBoxView.cpp

Page 2/3

```

67     } else if (id == GIRDER_3_BUTTON_ID) {
68         if (girder_3m->get_active()) {
69             worm->set_active(false);
70             girder_6m->set_active(false);
71         }
72     } else {
73         girder_3m->set_active(false);
74         worm->set_active(false);
75     }
76     disableMovingItems();
77     mode->set_active(false);
78     map_controller->addModeSignal(id);
79     leaveConsistent();
80     processing=false;
81 }
82 }
83
84 void ToolBoxView::enableMovingItems() {
85     turncw->set_sensitive(true);
86     turnccw->set_sensitive(true);
87     move->set_sensitive(true);
88     erase->set_sensitive(true);
89 }
90
91 void ToolBoxView::disableMovingItems() {
92     turncw->set_sensitive(false);
93     turnccw->set_sensitive(false);
94     move->set_sensitive(false);
95     erase->set_sensitive(false);
96 }
97
98 void ToolBoxView::changeMode() {
99     worm->set_sensitive(!mode->get_active());
100    girder_3m->set_sensitive(!mode->get_active());
101    girder_6m->set_sensitive(!mode->get_active());
102    if (!mode->get_active()){
103        disableMovingItems();
104    }
105    map_controller->changeModeSignal();
106 }
107
108 void ToolBoxView::leaveConsistent() {
109     if (!worm->get_active() && !girder_6m->get_active() &&
110         !girder_3m->get_active()){
111         processing=true;
112         worm->set_active(true);
113         map_controller->addModeSignal(WORM_BUTTON_ID);
114     }
115 }
116
117 void ToolBoxView::showSelected(int id) {
118     switch (id){
119     case WORM_BUTTON_ID:
120         selected->set (IMAGES_PATH+"/right_worm.png");
121         selected->show();
122         hideRotatingButtons();
123         break;
124     case GIRDER_3_BUTTON_ID:
125         selected->set (IMAGES_PATH+"Girder/girder_3_selected.png");
126         selected->show();
127         break;
128     case GIRDER_6_BUTTON_ID:
129         selected->set (IMAGES_PATH+"Girder/girder_6_selected.png");
130         selected->show();
131         break;
132     default:

```

jun 12, 18 20:48

## ToolBoxView.cpp

Page 3/3

```

133         hideSelected();
134         break;
135     }
136 }
137
138 void ToolBoxView::hideSelected() {
139     selected->hide();
140 }
141
142 void ToolBoxView::closeSelectionMode() {
143     disableMovingItems();
144     hideSelected();
145     mode->set_active(false);
146 }
147
148 void ToolBoxView::hideRotatingButtons() {
149     turncw->set_sensitive(false);
150     turnccw->set_sensitive(false);
151 }
152

```

jun 19, 18 14:51

## 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     // Deshabilita los botones de rotacion
40     void hideRotatingButtons();
41
42 public:
43     ToolBoxView(BaseObjectType *cobject,
44                 const Glib::RefPtr<Gtk::Builder> &builder);
45
46     // Se ejecuta cuando se selecciona un elemento de la zona "Agregar"
47     void onNewObjectClicked(unsigned int id);
48
49     // Habilita para el usuario la interacci3n con las acciones de la zona
50     // "Seleccion"
51     void enableMovingItems();
52
53     // Deshabilita para el usuario la interacci3n con las acciones de la zona
54     // "Seleccion"
55     void disableMovingItems();
56
57     // Enlaza la vista con el controlador
58     void bindController(MapController *controller);
59
60     // Alterna la vista entre el modo "Agregar" y modo "Seleccion"
61     void changeMode();
62
63     // Muestra el objeto seleccionado en el recuadro en la zona "Seleccion"
64     void showSelected(int id);
65
66     // Vac3a el recuadro en la zona "Seleccion"

```

jun 19, 18 14:51

## ToolBoxView.h

Page 2/2

```

67     void hideSelected();
68
69     // Sale del modo "Seleccion"
70     void closeSelectionMode();
71
72
73 };
74
75
76 #endif //WORMS_TOOLBOX_H

```

jun 10, 18 19:29

## 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
33 void WeaponView::resetAmmo() {
34     ammo_selector->set_sensitive(!default_checkbox_state);
35     ammo_selector->set_value(default_ammo_selector_value);
36     infinite->set_active(default_checkbox_state);
37 }
38
39 void WeaponView::bindController(WeaponController *controller) {
40     this->controller = controller;
41 }
42
43 const int WeaponView::getInitialAmmo() {
44     return default_checkbox_state ? -1 : default_ammo_selector_value;
45 }
46
47 void WeaponView::setAmmo(const int &ammo) {
48     if (ammo < 0) {
49         infinite->set_active(true);
50         ammo_selector->set_sensitive(false);
51     } else {
52         infinite->set_active(false);
53         ammo_selector->set_sensitive(true);
54         ammo_selector->set_value(ammo);
55     }
56 }

```

jun 07, 18 23:33

## 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 método.
27     void onAmmoValueChanged();
28
29     // Al cambiar el estado del checkbox se llama a este método.
30     void onCheckboxClicked();
31
32     // Muestra la munición predeterminada de esta arma
33     void resetAmmo();
34
35     // Enlaza la vista al controlador
36     void bindController(WeaponController *controller);
37
38     // Obtiene la munición inicial
39     const int getInitialAmmo();
40
41     // Establece la munición a mostrar
42     void setAmmo(const int &ammo);
43 };
44
45
46 #endif //WORMS_WEP_H

```

jun 26, 18 12:47

## Table of Content

Page 1/1

1	<b>Table of Contents</b>			
2	1	FileBoxController.cpp	sheets 1 to 1 ( 1 ) pages 1- 2	89 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	163 lines
5	4	MapController.h.....	sheets 4 to 4 ( 1 ) pages 7- 8	77 lines
6	5	UsablesController.cpp	sheets 5 to 5 ( 1 ) pages 9- 9	65 lines
7	6	UsablesController.h	sheets 5 to 5 ( 1 ) pages 10- 10	42 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	26 lines
13	12	FileReader.cpp.....	sheets 8 to 8 ( 1 ) pages 16- 16	51 lines
14	13	FileReader.h.....	sheets 9 to 9 ( 1 ) pages 17- 17	32 lines
15	14	FileWriter.cpp.....	sheets 9 to 10 ( 2 ) pages 18- 19	72 lines
16	15	FileWriter.h.....	sheets 10 to 10 ( 1 ) pages 20- 20	31 lines
17	16	InvalidMapError.cpp	sheets 11 to 11 ( 1 ) pages 21- 21	21 lines
18	17	InvalidMapError.h...	sheets 11 to 11 ( 1 ) pages 22- 22	24 lines
19	18	Map.cpp.....	sheets 12 to 12 ( 1 ) pages 23- 23	61 lines
20	19	Map.h.....	sheets 12 to 12 ( 1 ) pages 24- 24	46 lines
21	20	MapObject.cpp.....	sheets 13 to 13 ( 1 ) pages 25- 25	28 lines
22	21	MapObject.h.....	sheets 13 to 13 ( 1 ) pages 26- 26	29 lines
23	22	Weapon.cpp.....	sheets 14 to 14 ( 1 ) pages 27- 27	19 lines
24	23	Weapon.h.....	sheets 14 to 14 ( 1 ) pages 28- 28	25 lines
25	24	FileBoxView.cpp.....	sheets 15 to 15 ( 1 ) pages 29- 29	27 lines
26	25	FileBoxView.h.....	sheets 15 to 15 ( 1 ) pages 30- 30	30 lines
27	26	LifeView.cpp.....	sheets 16 to 16 ( 1 ) pages 31- 31	17 lines
28	27	LifeView.h.....	sheets 16 to 16 ( 1 ) pages 32- 32	26 lines
29	28	MapView.cpp.....	sheets 17 to 18 ( 2 ) pages 33- 35	163 lines
30	29	MapView.h.....	sheets 18 to 19 ( 2 ) pages 36- 37	78 lines
31	30	ToolBoxView.cpp.....	sheets 19 to 20 ( 2 ) pages 38- 40	153 lines
32	31	ToolBoxView.h.....	sheets 21 to 21 ( 1 ) pages 41- 42	77 lines
33	32	WeaponView.cpp.....	sheets 22 to 22 ( 1 ) pages 43- 43	57 lines
34	33	WeaponView.h.....	sheets 22 to 22 ( 1 ) pages 44- 44	47 lines