```
ExplosionView.cpp
May 31, 18 12:08
                                                                             Page 1/1
   #include "ExplosionView.h"
   #include <gtkmm/image.h>
   #include <glibmm/main.h>
   #include "Path.h"
   ExplosionView::ExplosionView(BulletView&& bullet) : bulletView(std::move(bullet)
        this->animation = Gdk::Pixbuf::create from file(EXPLOSION ANIMATION);
        int width = this->animation->get width();
        int height = this->animation->get height();
        for (int i = 0; i < height/width; i++) {
           Glib::RefPtr<Gdk::Pixbuf> aux = Gdk::Pixbuf::create_subpixbuf(this->anim
    ation, 0, i * width, width, width);
            this->animation_vector.push_back(aux);
12
13
14
        this->iter = this->animation_vector.begin();
15
16
   ExplosionView::~ExplosionView() {}
17
18
19
   ExplosionView::ExplosionView(ExplosionView&& other) :
            bulletView(std::move(other.bulletView)){
20
        this->animation vector = other.animation vector;
21
        this->animation = other.animation;
22
        this->iter = this->animation_vector.begin();
23
24
25
   bool ExplosionView::startCallBack() {
26
        Gtk::Image& image = (Gtk::Image&)this->bulletView.getWidget();
27
        image.set(*(this->iter));
28
        this->iter++;
29
        if (this->iter == this->animation_vector.end()) {
            this->bulletView.removeFromWorld();
31
32
            return false:
33
34
        return true;
35
36
   void ExplosionView::start() {
37
       Glib::signal_timeout().connect(sigc::mem_fun(*this, &ExplosionView::startCal
38
    1Back), 40);
39
   bool ExplosionView::hasFinished() {
        return this->iter == this->animation vector.end();
42
43
```

```
ExplosionView.h
May 31, 18 12:08
                                                                              Page 1/1
   #ifndef __CLIENTEXPLOSIONVIEW_H__
   #define ___CLIENTEXPLOSIONVIEW_H__
   #include <vector>
   #include <adkmm/pixbuf.h>
   #include "BulletView.h"
   /* Clase que se encarga de reproducir la animacion de una explosion */
   class ExplosionView {
       private:
           BulletView bulletView;
            std::vector<Glib::RefPtr<Gdk::Pixbuf>> animation_vector;
            Glib::RefPtr<Gdk::Pixbuf> animation;
14
            std::vector<Glib::RefPtr<Gdk::Pixbuf>>::iterator iter;
15
16
            /* Callback de start */
17
           bool startCallBack();
18
19
       public:
20
            /* Constructor */
21
            ExplosionView(BulletView&& bullet);
22
            /* Destructor */
23
24
            ~ExplosionView();
25
            /* Constructor por movimiento */
26
            ExplosionView(ExplosionView&& other);
27
28
29
            /* Realiza la animacion de la explosion */
30
            void start();
            /* Devuelve true si la animacion de la explosion finalizo */
33
            bool hasFinished();
34
35
   };
36
   #endif
```

#include "ExplosionViewList.h" ExplosionViewList::ExplosionViewList() {} ExplosionViewList::~ExplosionViewList() {} void ExplosionViewList::check() { std::list<ExplosionView>::iterator iter; iter = this->animations.begin(); a 10 while (iter != this->animations.end()) { 11 if (iter->hasFinished()) { 12 iter = this->animations.erase(iter); 13 } else { ++iter; 14 15 16 17 18 19 void ExplosionViewList::addAndStart(ExplosionView&& animation) { this->check(); 20

this->animations.push back(std::move(animation));

this->animations.back().start();

ExplosionViewList.cpp

May 31, 18 12:08

21

22

23 }

```
ExplosionViewList.h
May 31, 18 12:08
                                                                             Page 1/1
   #ifndef WORMS_EXPLOSIONVIEWLIST_H
   #define WORMS_EXPLOSIONVIEWLIST_H
   #include <liist>
   #include "ExplosionView.h"
   /* Clase que se encarga de almacenar animaciones de explosiones */
   class ExplosionViewList {
       private:
10
            std::list<ExplosionView> animations;
            /* Verifica si alguna animacion de la lista finalizo y las
13
            * elimina de la lista */
14
            void check();
15
16
       public:
17
            /* Constructor */
            ExplosionViewList();
18
19
20
            /* Destructor */
21
            ~ExplosionViewList();
22
23
24
            /* Agrega una animacion de explosion a la lista y la reproduce */
25
            void addAndStart (ExplosionView&& animation);
26
27
28
   #endif //WORMS_EXPLOSIONVIEWLIST_H
```

Page 1/1

```
WalkingAnimation.cpp
Jun 06. 18 20:08
                                                                                                                                                                                   Page 1/1
         #include "WalkingAnimation.h"
 2 #include "Path.h"
       #include "ObjectSizes.h"
         #define DIR RIGHT 1
 6
        #define DIR LEFT -1
 8
       WalkingAnimation::WalkingAnimation(Gtk::Image* worm image) : worm image(worm image)
                  dir(DIR RIGHT) {
                  this->walk image = Gdk::Pixbuf::create from file(WORMS PATH + "walk.png");
                  int width = this->walk_image->get_width();
                  int height = this->walk_image->get_height();
12
                  for (int i = 0; i < height / WORM_IMAGE_WIDTH; i++)</pre>
13
14
                           walk_queue.push(Gdk::Pixbuf::create_subpixbuf(this->walk_image, 0, i * W
         ORM IMAGE WIDTH, width, WORM IMAGE WIDTH));
 15
16
17
18
       WalkingAnimation::~WalkingAnimation() {}
19
        WalkingAnimation::WalkingAnimation(WalkingAnimation& other):
                  walk_queue(std::move(other.walk_queue)), walk_image(std::move(other.walk_ima
21
         ge)),
                  worm image(other.worm image), dir(other.dir) {}
22
23
         void WalkingAnimation::setMovementImage(char new_dir) {
24
                  if (new dir == this->dir) {
25
                           this->walk_queue.push(std::move(this->walk_queue.front()));
26
                           this->walk_queue.pop();
27
28
                  this->dir = new_dir;
29
30
                  this->setStaticImage();
31
32
33
        void WalkingAnimation::setStaticImage() {
                  this->worm_image->set(Gdk::Pixbuf::create_subpixbuf(this->walk_queue.back())
           WORM_IMAGE_WIDTH + this->dir * WORM_IMAGE_WIDTH, 0, WORM_IMAGE_WIDTH, WORM_IMAGE_WID
         E_WIDTH));
35
36
         void WalkingAnimation::updateWormImage(Gtk::Image* worm image)
                  this->worm image = worm image;
38
39
40
        char WalkingAnimation::getDir() const {
42
                  return this->dir;
43
```

```
WalkingAnimation.h
Jun 06. 18 20:08
                                                                             Page 1/1
    #ifndef WORMS_WALKINGANIMATION_H
   #define WORMS WALKINGANIMATION H
   #include <qtkmm/image.h>
   #include <qdkmm/pixbuf.h>
   #include <queue>
   /* Clase que se encarga de actualizar la imagen del worm al
     * moverse obteniendo una animacion del worm caminando */
   class WalkingAnimation {
        private:
            std::queue<Glib::RefPtr<Gdk::Pixbuf>> walk_queue;
            Glib::RefPtr<Gdk::Pixbuf> walk_image;
            Gtk::Image* worm_image;
15
            char dir:
16
17
        public:
            /* Constructor*/
18
19
            WalkingAnimation(Gtk::Image* worm_image);
20
21
            /* Destructor */
            ~WalkingAnimation();
23
            /* Constructor por movimiento */
24
25
            WalkingAnimation(WalkingAnimation&& other);
26
27
            /* Actualiza la imagen del worm por la siguiente
28
             * imagen del worm caminando */
29
            void setMovementImage(char new dir);
30
            /* Setea la imagen del worm por la imagen actual del
32
             * worm caminando */
33
            void setStaticImage();
34
35
36
            /* Devuelve la direccion del worm */
37
            char getDir() const;
            /* Actualiza el puntero de la imagen del worm */
39
            void updateWormImage(Gtk::Image* worm_image);
40
   };
41
   #endif //WORMS_WALKINGANIMATION_H
```

```
WeaponAnimation.cpp
Jun 06. 18 20:08
                                                                             Page 1/2
   #include "WeaponAnimation.h"
2 #include <glibmm/main.h>
   #include "WormView.h"
   #include "Path.h"
   #include "ObjectSizes.h"
   #include "WeaponNames.h"
   #define DIR RIGHT 1
10
   WeaponAnimation::WeaponAnimation(const std::string& weapon, Gtk::Image* worm ima
       worm_image(worm_image), angle(DEFAULT_ANGLE) {
12
       this->updateWeaponImage(weapon);
13
14
15
   WeaponAnimation::~WeaponAnimation() {}
   WeaponAnimation::WeaponAnimation(WeaponAnimation&& other):
17
       scope_vector(std::move(other.scope_vector)),
18
19
       scope_image(std::move(other.scope_image)),
20
       worm image (other.worm image),
       angle (other.angle) {}
22
   void WeaponAnimation::updateWeaponImage(const std::string& weapon) {
23
       this->scope vector.clear();
24
       this->scope image = Gdk::Pixbuf::create from file(WORMS PATH + weapon + " sc
25
   ope.png");
26
       int width = this->scope image->get width();
       int height = this->scope image->get height();
27
       for (int i = 0; i < height / WORM_IMAGE_WIDTH; i++) {</pre>
28
           this->scope_vector.push_back(Gdk::Pixbuf::create_subpixbuf(scope_image,
   0, i * WORM_IMAGE_WIDTH, width, WORM_IMAGE_WIDTH));
30
31
32
33
   void WeaponAnimation::changeWeapon(const std::string& weapon, char dir) {
       this->updateWeaponImage(weapon);
34
       this->setWeaponImage(dir);
35
36
37
   void WeaponAnimation::setWeaponImage(char dir) {
38
       int width = this->scope vector[(90 + this->angle) / 6]->get width() / 3;
       int height = this->scope vector[(90 + this->angle) / 6]->get height();
       this->worm_image->set(Gdk::Pixbuf::create_subpixbuf(this->scope_vector[(90 +
    this->angle) / 6], width + dir * width, 0, width, height));
42
43
   bool WeaponAnimation::batHitCallBack(std::vector<Glib::RefPtr<Gdk::Pixbuf>>::ite
   rator& iter, const int width, char dir) {
       this->worm_image->set(Gdk::Pixbuf::create_subpixbuf(*iter, 0, 0, width, WORM
    _IMAGE_WIDTH));
       ++iter:
       if (iter == this->scope_vector.end())
47
           this->updateWeaponImage(BAT_NAME);
           this->setWeaponImage(dir);
40
50
           return false;
51
52
       return true;
53
   void WeaponAnimation::weaponShootAnimation(const std::string &weapon, char dir)
55
56
       if (weapon != BAT NAME) {
           return;
57
58
       this->scope image = Gdk::Pixbuf::create from file(BAT HIT ANIMATION);
```

```
[75.42] Taller de programacion
                                WeaponAnimation.cpp
Jun 06. 18 20:08
                                                                            Page 2/2
        int width = this->scope_image->get_width() / 3;
        int height = this->scope_image->get_height();
        int pos x = width + dir * width;
62
        this->scope_vector.clear();
63
        for (int i = 0; i < height / WORM IMAGE WIDTH; i++) {
64
            this->scope vector.push back(Gdk::Pixbuf::create subpixbuf(scope image,
   pos_x, i * WORM_IMAGE_WIDTH, width, WORM_IMAGE_WIDTH));
        std::vector<Glib::RefPtr<Gdk::Pixbuf>>::iterator iter = this->scope vector.b
   egin();
        sigc::slot<bool> my slot = sigc::bind(sigc::mem fun(*this, &WeaponAnimation:
    :batHitCallBack), iter, width, dir);
       Glib::signal_timeout().connect(my_slot, 12);
70
71
72
   void WeaponAnimation::changeAngle(int angle, char dir) {
        this->angle = angle;
        this->setWeaponImage(dir);
74
75
   void WeaponAnimation::updateWormImage(Gtk::Image* worm image)
        this->worm image = worm image;
79
```

```
WeaponAnimation.h
Jun 06. 18 20:08
                                                                             Page 1/1
   #ifndef WORMS_WEAPONANIMATION_H
   #define WORMS WEAPONANIMATION H
   #include <qtkmm/image.h>
   #include <qdkmm/pixbuf.h>
   #include <vector>
   #include <string>
   class WormView;
9
10
11
   class WeaponAnimation {
12
13
            std::vector<Glib::RefPtr<Gdk::Pixbuf>> scope_vector;
           Glib::RefPtr<Gdk::Pixbuf> scope_image;
14
15
           Gtk::Image* worm_image;
16
           int angle;
17
            /* Actualiza las imagenes por las imagenes del arma nueva */
18
            void updateWeaponImage(const std::string& weapon);
19
20
21
            /* Callback */
           bool batHitCallBack(std::vector<Glib::RefPtr<Gdk::Pixbuf>>::iterator& it
22
   er, const int width, char dir);
23
24
       public:
            /* Constructor */
25
           WeaponAnimation(const std::string& weapon, Gtk::Image* worm_image);
26
27
            /* Destructor */
28
           ~WeaponAnimation();
29
30
            /* Constructor por movimiento */
31
           WeaponAnimation(WeaponAnimation&& other);
32
33
34
35
            /* Cambia la imagen del worm con el arma actual por una imagen
             * del worm con la nueva arma */
36
            void changeWeapon(const std::string& weapon, char dir);
37
38
            /* Setea la imagen del worm con el arma actual apuntando
39
             * con el angulo especifico */
40
            void setWeaponImage(char dir);
42
            /* Realiza la animacion del disparo del arma */
43
44
            void weaponShootAnimation(const std::string &weapon, char dir);
45
            /* Actualiza el angulo, cambiando la imagen del arma
46
47
             * por la correspondiente */
            void changeAngle(int angle, char dir);
48
49
            /* Actualiza el puntero de la imagen del worm */
50
            void updateWormImage(Gtk::Image* worm_image);
   };
52
53
   #endif //WORMS WEAPONANIMATION H
```

```
May 30, 18 20:03
                                         main.cpp
                                                                               Page 1/1
    #include <gtkmm/application.h>
   #include <qtkmm/window.h>
   #include "ServerMenu.h"
   #include "Path.h"
   int main(int argc, char* argv[]){
        auto app = Gtk::Application::create(argc, argv);
        Gtk::Window window;
        window.maximize();
10
        window.set title(CLIENT WINDOW NAME);
12
13
        window.set_icon_from_file(ICON_PATH);
14
15
        ServerMenu server menu (window):
16
17
        app->run (window);
18
19
        return 0;
20
```

Jun 01, 18 13:12 ButtonBuilder.cpp Page 1/1 #include "ButtonBuilder.h" #include <string> #include <gtkmm/label.h> #include <gdkmm/rgba.h> void ButtonBuilder::buildButton(Gtk::Button* button) { std::string text = button->get_label(); Gtk::Label* label = (Gtk::Label*)button->get_child(); label->set_markup("
 label->override_color(Gdk::RGBA("black")); }

```
Jun 03, 18 12:56

ButtonBuilder.h

#ifindef WORMS_BUTTONBUILDER_H
#define WORMS_BUTTONBUILDER_H

#include <gtkmm/button.h>

class ButtonBuilder {
    public:
        /* Modifica la visualización del label del boton */
        static void buildButton(Gtk::Button* button);

10    };

11    #endif //WORMS_BUTTONBUILDER_H
```

```
CreateGameMenu.cpp
Jun 05. 18 15:19
                                                                               Page 1/1
   #include "CreateGameMenu.h"
#include <qtkmm/builder.h>
   #include <qlibmm/main.h>
   #include "Path.h"
   #include "GamePlayers.h"
   #include "ButtonBuilder.h"
   CreateGameMenu::CreateGameMenu(Gtk::Window& window, MenuView& first menu, Client
    Protocol& protocol, std::string&& name, int quantity):
        SelectableListMenu(window, first menu, protocol, std::move(name)) {
10
11
        Glib::RefPtr<Gtk::Builder> builder = Gtk::Builder::create_from_file(GLADE_PA
   TH + "client_CreateGameMenu.glade");
12
13
        builder->get_widget("error", this->error);
14
        builder->get_widget("game_name", this->game_name);
        builder->get_widget("players_number", this->players_number);
15
        builder->get_widget("games", this->games);
16
17
        builder->get_widget("quit_game", this->quit_game);
18
19
        this->configure (quantity);
20
        ButtonBuilder::buildButton(quit_game);
21
22
23
        builder->get widget("create game menu", this->menu);
24
25
        this->addMenu();
        this->quit game->signal clicked().connect(sigc::mem fun(*this, &CreateGameMe
26
    nu::quitButtonPressed));
27
28
   CreateGameMenu::~CreateGameMenu() { }
    void CreateGameMenu::selectButtonPressed(Glib::ustring map_name) {
31
        std::string name(this->game_name->get_text());
32
33
        if (name.empty()) {
            this->error->set_label("Debe ingresar el nombre de la partida");
34
35
36
37
        size t players = this->players number->get value as int();
38
        if (players < min players || players > max players) {
            std::string message("El numero de jugadores debe estar entre ");
40
            message += std::to_string(min_players) + std::string("y") + std::to_str
41
    ing(max players);
            this->error->set_label(message);
42
43
            return;
44
45
        try {
46
            this->protocol.sendString(map name);
47
            this->protocol.sendString(name);
            this->protocol.sendLength(players);
49
            bool result = this->protocol.receiveChar();
50
            if (!result) {
51
52
                this->showErrorAndRestart ("Ocurrio un error al crear la partida");
53
                this->waitToPlayers();
54
55
          catch (const SocketException& e) {
56
            this->showFatalError();
57
59
```

```
CreateGameMenu.h
Jun 01. 18 13:12
                                                                              Page 1/1
    #ifndef ___CREATEGAMEMENU___
   #define ___CREATEGAMEMENU__
   #include <gtkmm/button.h>
   #include <gtkmm/entry.h>
   #include <gtkmm/spinbutton.h>
   #include "SelectableListMenu.h"
   /* Clase que se encarga de los pasos necesarios para que el
     * jugador cree una partida */
   class CreateGameMenu: public SelectableListMenu{
            Gtk::Entry* game_name;
14
            Gtk::SpinButton* players_number;
15
            Gtk::Button* quit game;
16
17
            /* Handler del boton de seleccion */
            void selectButtonPressed(Glib::ustring map_name) override;
18
19
20
        public:
21
            /* Constructor */
            CreateGameMenu (Gtk::Window& window, MenuView& first menu, ClientProtocol
   & protocol, std::string&& name, int quantity);
23
            /* Destructor */
24
            ~CreateGameMenu();
25
26
28
   #endif
```

```
GameMenu.cpp
Jun 03. 18 12:56
                                                                              Page 1/2
   #include "GameMenu.h"
#include <qtkmm/builder.h>
   #include "Path.h"
   #include "CreateGameMenu.h"
   #include "JoinGameMenu.h"
   #include "ButtonBuilder.h"
   GameMenu::GameMenu(Gtk::Window& window, ClientProtocol& protocol):
        MenuView(window, *this, protocol) {
10
        Glib::RefPtr<Gtk::Builder> builder = Gtk::Builder::create from file(GLADE PA
   TH + "client GameMenu.glade");
12
        builder->get_widget("error", this->error);
        builder->get_widget("player_name", this->player_name);
13
14
15
        builder->get widget("game menu", this->menu);
16
        this->addMenu();
17
18
10
        Gtk::Button *create game, *join game, *quit game;
20
        builder->get widget("create game", create game);
21
        builder->get_widget("join_game", join_game);
22
        builder->get_widget("quit_game", quit_game);
23
24
        ButtonBuilder::buildButton(create game);
25
        ButtonBuilder::buildButton(join_game);
26
        ButtonBuilder::buildButton(quit game);
27
28
        create_game->signal_clicked().connect(sigc::mem_fun(*this, &GameMenu::create
29
   ButtonPressed));
        join_game->signal_clicked().connect(sigc::mem_fun(*this, &GameMenu::joinButt
        quit_game->signal_clicked().connect(sigc::mem_fun(*this, &GameMenu::quitButt
31
    onPressed));
32
33
   GameMenu::~GameMenu(){}
34
35
   void GameMenu::createButtonPressed() {
36
        if (this->selectAction(CREATE GAME ACTION)) {
37
            std::string name(this->player name->get text());
            int quantity = this->protocol.receiveLength();
39
            if (quantity == 0) {
40
                this->showErrorAndRestart ("No hay mapas para crear una partida");
41
42
              else
                this->next_menu = std::unique_ptr<MenuView>(new CreateGameMenu(this-
    >window, *this, this->protocol, std::move(name), quantity));
44
45
46
   void GameMenu::joinButtonPressed() {
48
       if (this->selectAction(JOIN_GAME_ACTION)) {
49
            std::string name(this->player_name->get_text());
50
51
            int quantity = this->protocol.receiveLength();
52
            if (quantity == 0) {
                this->showErrorAndRestart ("No hay partidas disponibles");
53
              else
54
                this->next_menu = std::unique_ptr<MenuView>(new JoinGameMenu(this->w
55
           *this, this->protocol, std::move(name), quantity));
    indow,
57
58
   bool GameMenu::selectAction(char action) {
```

```
GameMenu.cpp
Jun 03. 18 12:56
                                                                                Page 2/2
        std::string name(this->player_name->get_text());
62
        if (name.empty()) {
            this->error->set label("Debe ingresar su nombre");
63
            return false:
64
65
66
        try{
            this->protocol.sendChar(action);
67
68
            this->protocol.sendString(name);
69
            this->window.remove();
70
            return true;
        } catch (const SocketException& e) {
72
            this->showFatalError();
73
            return false;
74
75
```

GameMenuField.cpp Jun 02. 18 12:18 Page 1/1 #include "GameMenuField.h" #include <qdkmm/rqba.h> #include "Path.h" #include "ButtonBuilder.h" 6 GameMenuField::GameMenuField(const std::string& title): container(true, 20) { size t extension = title.rfind(YAML EXTENSION); this->title.set_markup(title.substr(0, extension)); this->title.override_color(Gdk::RGBA("black")); 10 this->title.override background color(Gdk::RGBA("white")); this->container.pack start(this->title); 12 this->container.pack_end(this->button); 13 this->button.set_label("Seleccionar"); 14 15 ButtonBuilder::buildButton(&this->button); 16 17 GameMenuField::~GameMenuField() { } 18 19 20 GameMenuField::GameMenuField(GameMenuField&& other): title(std::move(other.title 21 button(std::move(other.button)), container(std::move(other.container)){} 22 Gtk::Container& GameMenuField::getContainer() { 23 return this->container; 24 25 Gtk::Button& GameMenuField::getButton() { 27 return this->button; 28 29 }

```
GameMenuField.h
May 28, 18 18:21
                                                                              Page 1/1
   #ifndef ___GAMEMENUFIELD_H__
   #define ___GAMEMENUFIELD_H__
   #include <gtkmm/hvbox.h>
   #include <gtkmm/label.h>
   #include <qtkmm/button.h>
   #include <string>
   class GameMenuField{
       private:
            Gtk::Label title;
            Gtk::Button button;
13
            Gtk:: HBox container;
14
15
       public:
16
            /* Constructor */
17
            GameMenuField(const std::string& title);
18
            /* Destructor */
19
20
            ~GameMenuField();
21
            /* Constructor por movimiento */
            GameMenuField(GameMenuField&& other);
23
24
25
26
            /* Devuelve el contenedor del menu */
            Gtk::Container& getContainer();
27
28
            /* Devuelve el boton del menu */
29
            Gtk::Button& getButton();
30
   };
31
34 #endif
```

```
GameMenu.h
May 31, 18 13:58
                                                                              Page 1/1
   #ifndef ___GAMEMENU__
   #define ___GAMEMENU___
   #include <gtkmm/button.h>
   #include <gtkmm/entry.h>
   #include <gtkmm/window.h>
   #include <string>
   #include <memory>
   #include "ClientProtocol.h"
   #include "MenuView.h"
   /* Clase que se encarga de controlar el menu del juego */
   class GameMenu: public MenuView{
       private:
14
15
            Gtk::Entry* player_name;
16
17
            /* Crea el boton de creacion de partida */
            void createButtonPressed();
18
19
20
            /* Crea el boton de unirse a partida */
21
            void joinButtonPressed();
22
            /* Envia la accion implementada */
23
            bool selectAction (char action);
24
25
        public:
26
            /* Constructor */
27
            GameMenu(Gtk::Window& window, ClientProtocol& protocol);
28
29
            /* Destructor */
30
            ~GameMenu();
31
32
   };
33
   #endif
```

```
JoinGameMenu.cpp
Jun 03. 18 12:56
                                                                                Page 1/1
    #include "JoinGameMenu.h"
   #include <qtkmm/builder.h>
   #include <qlibmm/main.h>
   #include "Path.h"
   #include "WaitingLabel.h"
   #include "ButtonBuilder.h"
   JoinGameMenu::JoinGameMenu(Gtk::Window& window, MenuView& first menu, ClientProt
   ocol& protocol, std::string&& name, int quantity):
        SelectableListMenu(window, first menu, protocol, std::move(name)) {
        Glib::RefPtr<Gtk::Builder> builder = Gtk::Builder::create_from_file(GLADE_PA
   TH + "client_JoinGameMenu.glade");
12
        builder->get_widget("error", this->error);
13
14
        builder->get widget("game", this->games);
15
        builder->get_widget("quit_game", this->quit_game);
16
17
        this->configure (quantity);
18
19
        ButtonBuilder::buildButton(quit game);
20
        builder->get_widget("join_game_menu", this->menu);
21
22
23
        this->addMenu();
24
        quit_game->signal_clicked().connect(sigc::mem_fun(*this, &JoinGameMenu::quit
   ButtonPressed));
26
27
   JoinGameMenu::~JoinGameMenu() { }
31
   void JoinGameMenu::selectButtonPressed(Glib::ustring game_name) {
32
33
34
            this->protocol.sendString(game_name);
            bool result = this->protocol.receiveChar();
35
            if (!result) {
36
                this->showErrorAndRestart ("Ocurrio un error al unirse a la partida");
37
38
              else ·
                this->waitToPlayers();
        } catch (const SocketException& e) {
41
42
            this->showFatalError();
43
44
```

```
JoinGameMenu.h
Jun 01. 18 13:12
                                                                             Page 1/1
   #ifndef ___JOINGAMEMENU__
   #define ___JOINGAMEMENU___
   #include "SelectableListMenu.h"
    /* Clase que se encarga de los pasos necesarios para que el
    * jugador se una a una partida */
   class JoinGameMenu: public SelectableListMenu{
10
       private:
            Gtk::Button* quit game;
12
13
            /* Handler del boton de unirse a partida */
            void selectButtonPressed(Glib::ustring game_name) override;
14
15
16
       public:
17
            /* Constructor */
            JoinGameMenu(Gtk::Window& window, MenuView& first_menu, ClientProtocol&
18
   protocol, std::string&& name, int quantity);
20
            /* Destructor */
21
            ~JoinGameMenu();
22
23
   #endif
```

```
MenuView.cpp
Jun 03. 18 12:56
                                                                             Page 1/1
   #include "MenuView.h"
   #include "ServerFatalError.h"
   MenuView::MenuView(Gtk::Window& window, MenuView& main menu, ClientProtocol& pro
        window(window), protocol(protocol), main menu(main menu) {
        Glib::RefPtr<Gdk::Pixbuf> aux = Gdk::Pixbuf::create from file(BACKGROUND MEN
   U IMAGE);
        this->background.set(aux);
        this->menu container.add overlay (this->background);
10
12
   MenuView::~MenuView() {
13
        delete this->menu:
14
   void MenuView::showFatalError() {
        ServerFatalError error (this->window);
18
19
   void MenuView::showErrorAndRestart(const std::string& error) {
        this->window.remove();
21
        this->main_menu.showError(error);
22
23
        this->window.add(this->main menu.menu container);
24
   void MenuView::showError(const std::string& error) {
26
        this->error->set_label(error);
27
28
29
   void MenuView::quitButtonPressed() {
        this->window.close();
31
32
33
34
   void MenuView::addMenu()
        this->menu_container.add_overlay(*this->menu);
        this->window.add(this->menu_container);
36
        this->window.show_all();
37
38 }
```

```
MenuView.h
Jun 05. 18 14:28
                                                                             Page 1/1
   #ifndef __MENUVIEW_H__
2 #define MENUVIEW H
   #include <atkmm/hvbox.h>
   #include <gtkmm/label.h>
   #include <qtkmm/window.h>
   #include <gtkmm/overlay.h>
   #include <gtkmm/image.h>
   #include <memorv>
   #include "ClientProtocol.h"
   class MenuView{
13
       private:
14
            /* Muestra un mensaje de error */
15
            void showError (const std::string& error);
16
17
       protected:
            Gtk::Window& window;
18
19
            ClientProtocol& protocol:
20
            Gtk::Label* error:
21
            std::unique ptr<MenuView> next menu;
22
            MenuView& main menu;
           Gtk::Container* menu;
23
24
25
            Gtk::Overlay menu container;
            Gtk:: Image background;
26
27
            /* Muestra un mensaje de error y cierra la aplicacion*/
28
            void showFatalError();
29
30
            /* Muestra un mensaje de error v reinicia */
31
            void showErrorAndRestart(const std::string& error);
32
33
            /* Agrega el menu al world y el world al window */
34
            void addMenu();
35
36
            /* Handler del boton de salir */
37
            void quitButtonPressed();
38
39
       public:
40
41
           MenuView(Gtk::Window& window, MenuView& main menu, ClientProtocol& proto
42
   col);
43
44
            /* Destructor */
            virtual ~MenuView();
45
46
   };
47
   #endif
```

```
SelectableListMenu.cpp
Jun 01. 18 14:20
                                                                             Page 1/1
   #include "SelectableListMenu.h"
   SelectableListMenu::SelectableListMenu(Gtk::Window& window, MenuView& first menu
   , ClientProtocol& protocol, std::string&& name):
       MenuView(window, first menu, protocol), player name(std::move(name)){}
   SelectableListMenu::~SelectableListMenu(){}
   void SelectableListMenu::configure(int quantity) {
            for (int i = 0; i < quantity; i++) {
                std::string field = this->protocol.receiveString();
                this->addField(field);
12
13
14
        }catch (const SocketException& e) {
15
            this->showFatalError();
16
17
18
        for (auto it = this->fields.begin(); it != this->fields.end(); ++it) {
19
            this->games->pack start(it->getContainer());
20
21
        this->games->show();
22
23
   void SelectableListMenu::addField(const std::string& field name) {
        GameMenuField field(field name);
25
        this->fields.push_back(std::move(field));
26
        this->fields.back().getButton().signal clicked().connect(sigc::bind<Glib::us
   tring>(sigc::mem_fun(*this,
28
                       &SelectableListMenu::selectButtonPressed), field name));
   bool SelectableListMenu::createPlayer() {
31
32
            this->player = std::unique_ptr<Player>(new Player(std::move(this->protoc
33
   ol), this->player name, this->window));
        } catch (const std::exception& e) {
            this->showFatalError();
35
36
        return false;
   void SelectableListMenu::waitToPlayers() {
        this->window.remove();
        this->window.add(this->waiting_label.getWidget());
42
        this->window.show_all();
        sigc::slot<bool> my_slot = sigc::mem_fun(*this, &SelectableListMenu::createP
   layer);
        Glib::signal idle().connect(mv slot);
46
```

SelectableListMenu.h May 31, 18 13:56 Page 1/1 #ifndef ___SELECTABLELISTMENU_H__ 2 #define __SELECTABLELISTMENU_H__ #include <gtkmm/hvbox.h> #include <gtkmm/label.h> #include <qtkmm/window.h> #include <memory> #include <string> #include <vector> 10 #include "ClientProtocol.h" #include "MenuView.h" 12 #include "WaitingLabel.h" #include "Player.h" #include "GameMenuField.h" 15 16 class SelectableListMenu: public MenuView{ 17 protected: Gtk::Box* games; 18 std::string player_name; 19 WaitingLabel waiting_label; 20 21 std::vector<GameMenuField> fields; 22 std::unique ptr<Player> player; 23 /* Realiza la configuracion del juego */ 24 25 void configure(int quantity); 26 /* Agrega un campo a la lista */ 27 void addField(const std::string& field name); 28 29 /* Crea un nuevo jugador */ 30 bool createPlayer(); 31 /* Handler del boton de seleccion */ 33 virtual void selectButtonPressed(Glib::ustring field_name) = 0; 34 35 36 /* Muestra el mensaje esperando jugadores */ 37 void waitToPlayers(); 38 public: 39 /* Constructor */ 40 SelectableListMenu(Gtk::Window& window, MenuView& first menu, ClientProt 41 ocol& protocol, std::string&& name); 42 /* Destructor */ 43 44 ~SelectableListMenu(); 45 47 #endif

```
Jun 01, 18 13:45

ServerFatalError.cpp

#include "ServerFatalError.h"
#include <gtkmm/messagedialog.h>

ServerFatalError::ServerFatalError(Gtk::Window& window) {
    Gtk::MessageDialog dialog(window, "Ocurrio un error con la conexion del servidor", false, Gtk::MESSAGE_ERROR, Gtk::BUTTONS_CLOSE, true);
    dialog.run();
    window.close();
    }
}

ServerFatalError::~ServerFatalError(){}
```

```
ServerMenu.cpp
Jun 05. 18 14:27
                                                                               Page 1/2
    #include "ServerMenu.h"
   #include <atkmm/builder.h>
   #include "Path.h"
   #include "ButtonBuilder.h"
   ServerMenu::ServerMenu(Gtk::Window& window): window(window) {
        Glib::RefPtr<Gtk::Builder> builder = Gtk::Builder::create from file(GLADE PA
   TH + "client ServerMenu.glade");
        builder->get widget("error", this->error);
        builder->get widget("host", this->host);
        builder->get_widget("service", this->service);
12
        builder->get_widget("connect", this->connect);
13
        builder->get_widget("quit_game", this->quit);
14
15
        ButtonBuilder::buildButton(this->quit);
16
        ButtonBuilder::buildButton(this->connect);
17
18
        builder->get_widget("server_menu", this->menu);
19
20
        builder->get widget("background", this->background);
        Glib::RefPtr<Gdk::Pixbuf> aux = Gdk::Pixbuf::create from file(BACKGROUND MEN
21
   U IMAGE);
        this->background->set(aux);
22
23
        this->window.add(*this->menu);
24
        this->window.show_all();
25
26
        this->connect->signal_clicked().connect(sigc::mem_fun(*this, &ServerMenu::co
   nnectButtonPressed));
        this->quit->signal_clicked().connect(sigc::mem_fun(*this, &ServerMenu::quitB
   uttonPressed));
29
30
   ServerMenu::~ServerMenu() {
31
32
        delete this->menu;
33
   void ServerMenu::connectButtonPressed() {
35
        std::string host(this->host->get_text());
36
        if (host.empty()) {
            this->error->set label("Debe ingresar un host");
            return;
39
40
41
        std::string service(this->service->get_text());
42
43
        if (service.empty()){
            this->error->set_label("Debe ingresar un servicio");
44
45
            return;
46
47
        this->connectToServer(host, service);
49
51
   void ServerMenu::quitButtonPressed() {
        this->window.close();
52
53
   void ServerMenu::connectToServer(const std::string &host, const std::string &ser
55
   vice) {
        try{
56
            Socket socket(Socket::Client(host.c_str(), service.c_str()));
            this->protocol.reset(new ClientProtocol(std::move(socket), this->window)
   );
            this->window.remove();
59
            this->next_menu = std::unique_ptr<MenuView>(new GameMenu(this->window,
```

```
        Jun 05, 18 14:27
        ServerMenu.cpp
        Page 2/2
```

```
ServerMenu.h
Jun 05. 18 14:28
                                                                             Page 1/1
   #ifndef ___SERVERMENU__
   #define __SERVERMENU_
   #include <gtkmm/application.h>
   #include <gtkmm/hvbox.h>
   #include <qtkmm/button.h>
   #include <gtkmm/entry.h>
   #include <qtkmm/label.h>
   #include <gtkmm/window.h>
10 #include <gtkmm/overlay.h>
11 #include <gtkmm/image.h>
12 #include <string>
13 #include <memory>
14 #include "ClientProtocol.h"
   #include "GameMenu.h"
   #include "MenuView.h"
   /* Menu de conexion con el servidor */
   class ServerMenu{
20
       private:
21
            Gtk::Label* error;
            Gtk::Entry* host;
            Gtk::Entry* service;
23
            Gtk::Button* connect;
24
25
            Gtk::Button* quit;
26
            Gtk::Window& window;
            Gtk::Container* menu;
27
            std::unique_ptr<MenuView> next_menu;
28
            std::unique_ptr<ClientProtocol> protocol;
29
            Gtk::Image* background;
30
31
            /* Handler del boton de conexion */
            void connectButtonPressed();
33
34
            /* Handler del boton de salir */
35
            void quitButtonPressed();
36
37
            /* Intenta realizar una conexion con el servidor */
38
            void connectToServer(const std::string &host, const std::string &service
39
   );
40
        public:
            /* Constructor */
42
            ServerMenu (Gtk::Window& window);
43
44
            /* Destructor */
45
            ~ServerMenu();
47
   };
49 #endif
```

WaitingLabel.cpp May 27, 18 22:46 #include "WaitingLabel.h" const std::string begining(""); const std::string ending(""); WaitingLabel::WaitingLabel() { 6 this->label.set use markup(true); this->label.set_markup(begining + "Esperando jugadores..." + ending); 8 this->label.show(); 9 10 } 12 WaitingLabel::~WaitingLabel(){} 13 Gtk::Widget& WaitingLabel::getWidget(){ 14 15 return this->label; 16 }

```
WaitingLabel.h
Jun 03, 18 12:56
                                                                            Page 1/1
   #ifndef __WAITINGLABEL_H__
   #define ___WAITINGLABEL_H__
   #include <gtkmm/label.h>
   /* Label de que indica la espera a otros jugadores */
   class WaitingLabel{
       private:
            Gtk::Label label;
       public:
            /* Constructor */
            WaitingLabel();
13
14
15
            /* Destructor */
            ~WaitingLabel();
16
17
18
            /* Devuelve el contenedor del mensaje */
19
            Gtk::Widget& getWidget();
20
   };
21
23 #endif
```

Page 1/1

```
MusicPath.h
Jun 03. 18 12:56
                                                                                  Page 1/1
    #ifndef WORMS_MUSICPATH_H
   #define WORMS MUSICPATH H
    #include <string>
    #include "Path.h"
    const std::string BACKGROUND_MUSIC = SOUNDS_PATH + "BackgroundMusic.mp3";
   const std::string START TURN SOUND = SOUNDS PATH + "Misc/StartRound.wav";
   const std::string TICK SOUND = SOUNDS PATH + "Misc/TimerTick,way";
10 const std::string RUN AWAY SOUND = SOUNDS PATH + "Worms/RunAway.wav";
11 const std::string DEATH SOUND = SOUNDS PATH + "Worms/Death.way";
   const std::string DAMAGE_RECEIVE_SOUND = SOUNDS_PATH + "Worms/DamageReceive.wav";
   const std::string EXPLOSION_SOUND = SOUNDS_PATH + "Weapons/Explosion.wav";
const std::string TELEPORT_SOUND = SOUNDS_PATH + "Weapons/Teleportation.wav";
   const std::string BAT_SOUND = SOUNDS_PATH + "Weapons/BaseballSound.wav";
    const std::string HOLY_GRENADE_SOUND = SOUNDS_PATH + "Weapons/HolyGrenade.wav";
   const std::string AIR_ATTACK_SOUND = SOUNDS_PATH + "Weapons/AirAttack.wav";
   const std::string SHOOT_SOUND = SOUNDS_PATH + "Weapons/ShootWeapon.wav";
18
   const std::string ROLLBACK_SOUND = SOUNDS_PATH + "Misc/RollBack.wav";
   const std::string JUMP_SOUND = SOUNDS_PATH + "Misc/Jump.wav";
   const std::string SELECT WEAPON SOUND = SOUNDS PATH + "Misc/SelectWeapon.wav";
   const std::string NO AMMO SOUND = SOUNDS PATH + "Misc/NoAmmo.wav";
   const std::string VICTORY SOUND = SOUNDS PATH + "Worms/Victory.WAV";
   #endif //WORMS MUSICPATH H
```

```
MusicPlayer.cpp
Jun 03. 18 12:56
                                                                                 Page 1/3
    #include "MusicPlayer.h"
   #include "MusicPlayerException.h"
   #include "WeaponNames.h"
    #include "Protocol.h"
    #include "MusicPath.h"
   MusicPlayer::MusicPlayer() {
        this -> music = NULL;
        // Initialize SDL.
10
        if (SDL Init(SDL INIT AUDIO) < 0) {</pre>
            throw MusicPlayerException ("Error al inicializar SDL");
12
13
14
        //Initialize SDL_mixer
15
        if (Mix_OpenAudio( 22050, MIX_DEFAULT_FORMAT, 2, 4096) == -1) {
16
            throw MusicPlayerException ("Error al inicializar SDL mixer");
17
18
19
        // Load background music
20
        this->music = Mix_LoadMUS(BACKGROUND_MUSIC.c_str());
21
        if (this->music == NULL) {
23
24
25
   MusicPlayer::~MusicPlayer() {
        Mix HaltChannel (-1);
26
        this->stop();
27
        if (this->music != NULL)
28
            Mix FreeMusic(this->music);
29
30
        std::map<int, Mix Chunk*>::iterator iter;
31
        for (iter = this->effects.begin(); iter != this->effects.end(); iter++) {
32
            Mix_FreeChunk(iter->second);
33
34
        // quit SDL_mixer
35
        Mix_CloseAudio();
36
        Mix Ouit();
37
        SDL Quit();
38
39
40
   void MusicPlayer::check(int channel) {
        if (this->effects.find(channel) != this->effects.end()) {
            // elimino el audio anterior de este canal
43
            Mix FreeChunk (this->effects.at (channel));
44
45
            this->effects.erase(channel);
46
        std::map<int, Mix_Chunk*>::iterator iter = this->effects.begin();
47
        while (iter != this->effects.end()) {
            if (!Mix_Playing(iter->first)) {
49
                Mix FreeChunk (iter->second);
50
                 iter = this->effects.erase(iter);
52
            } else {
53
                iter++;
54
55
56
   void MusicPlayer::addEffect(const std::string& audio) {
        int channel;
        Mix Chunk* effect = NULL;
60
        effect = Mix_LoadWAV(audio.c_str());
61
62
        if (effect == NULL) {
63
            return:
64
        if ((channel = Mix_PlayChannel(-1, effect, 0)) == -1) {
65
            Mix FreeChunk (effect):
```

```
MusicPlayer.cpp
Jun 03. 18 12:56
                                                                               Page 2/3
            return:
68
        this->check(channel);
69
        this->effects.insert(std::make pair(channel, effect));
70
71
72
73
    void MusicPlayer::playMusic() {
        Mix PlayMusic (this->music, -1);
74
        Mix VolumeMusic (MIX MAX VOLUME / 4);
75
76
77
    void MusicPlayer::playStartTurnSound()
        this->addEffect (START_TURN_SOUND);
79
80
81
82
    void MusicPlayer::playTickSound()
83
        this->addEffect (TICK_SOUND);
84
85
86
    void MusicPlayer::playDeathSound() {
87
        this->addEffect (DEATH SOUND);
88
    void MusicPlayer::playDamageReceiveSound() {
90
91
        this->addEffect (DAMAGE RECEIVE SOUND);
92
93
    void MusicPlayer::playExplosionSound(const std::string& weapon) {
94
        if (weapon == HOLY_GRENADE_NAME)
95
            this->addEffect(HOLY GRENADE SOUND);
96
            this->addEffect(EXPLOSION_SOUND);
99
100
101
    void MusicPlayer::playVictory()
102
        this->addEffect (VICTORY_SOUND);
103
104
105
    void MusicPlayer::playNoAmmo() {
106
        this->addEffect(NO AMMO SOUND);
107
108
109
    void MusicPlayer::stop() {
110
111
       Mix HaltMusic();
112
113
    void MusicPlayer::playWeaponShotSound(const std::string& weapon) {
114
        if (weapon == TELEPORT_NAME) {
115
            this->addEffect (TELEPORT_SOUND);
116
        } else if (weapon == BAT NAME) {
117
            this->addEffect (BAT_SOUND);
          else if (weapon == DYNAMITE_NAME)
119
            this->addEffect (RUN_AWAY_SOUND);
120
          else if (weapon == AIR_ATTACK_NAME)
121
            this->addEffect (AIR ATTACK SOUND);
122
123
            this->addEffect(SHOOT_SOUND);
124
125
126
127
    void MusicPlayer::playJumpSound(char action) {
128
        if (action == ROLLBACK) {
129
            this->addEffect (ROLLBACK_SOUND);
130
        } else if (action == JUMP) {
131
            this->addEffect(JUMP_SOUND);
132
```

```
MusicPlayer.cpp
Jun 03. 18 12:56
                                                                             Page 3/3
134
135
   void MusicPlayer::playSelectWeaponSound()
        this->addEffect (SELECT WEAPON SOUND);
137
138
```



```
MusicPlayerException.h
May 22, 18 11:29
                                                                           Page 1/1
   #ifndef __MUSICPLAYEREXCEPTION_H__
   #define __MUSICPLAYEREXCEPTION_H__
   #include <exception>
   #include <string>
   class MusicPlayerException: public std::exception{
       private:
           std::string msg;
10
12
           //Crea la excepcion
13
           explicit MusicPlayerException(std::string msg);
14
15
           //Destruye la excepcion
16
           virtual ~MusicPlayerException();
17
           //Devuelve el mensaje de error
18
           virtual const char* what() const noexcept;
19
20
   };
21
22 #endif
```

```
MusicPlayer.h
Jun 03. 18 12:56
                                                                                Page 1/2
   #ifndef __MUSICPLAYER_H__
2 #define __MUSICPLAYER_H_
   #include <SDL2/SDL.h>
   #include <SDI2/SDI mixer.h>
   #include <map>
   #include <string>
    /* Clase que se enecarga de reproducir musica y efectos
    * de sonido */
   class MusicPlayer {
       private:
            Mix_Music* music; // Musica de fondo
13
            std::map<int, Mix_Chunk*> effects;
14
15
            /* Verifica si algunos efectos de la lista finalizaon y los
* libera. AdemÃ;s libera el efecto que se encuentre guardado
16
17
             * en la lista con clave channel */
18
            void check(int channel);
19
20
21
            /* Agrega un nuevo efecto a la lista y lo reproduce */
22
            void addEffect(const std::string& audio);
23
        public:
24
25
            /* Constructor */
            MusicPlayer();
26
27
            /* Destructor */
28
            ~MusicPlayer();
29
30
            /* Reproduce la musica de fondo */
31
            void playMusic();
32
33
            /* Reproduce el sonido de inicio de turno */
34
            void playStartTurnSound();
35
36
            /* Reproduce el sonido de falta de tiempo */
37
            void playTickSound();
38
39
            /* Reproduce el sonido de muerte de un worm */
40
            void playDeathSound();
41
42
            /* Reproduce el sonido de daño recibido */
43
            void playDamageReceiveSound();
44
45
            /* Reproduce el sonido de la explosion */
46
            void playExplosionSound(const std::string& weapon);
47
48
            /* Reproduce el sonido de arma disparada */
49
            void playWeaponShotSound(const std::string& weapon);
50
            /* Reproduce el sonido de salto o rollback */
52
            void playJumpSound(char action);
53
54
            /* Reproduce el sonido de arma seleccionada */
55
56
            void playSelectWeaponSound();
57
            /* Reproduce el sonido de victoria */
58
            void playVictory();
59
60
            /* Reproduce el sonido de arma descargada */
61
            void playNoAmmo();
62
63
            /* Detiene la reproduccion de la musica de fondo */
64
            void stop();
65
66 };
```

Jun 03, 18 12:56	MusicPlayer.h	Page 2/2
67 68		
69 #endif		

```
Plaver.cpp
Jun 06. 18 20:08
                                                                             Page 1/2
   #include "Player.h"
2 #include "WeaponNames.h"
   Player::Player(ClientProtocol protocol. const std::string& name. Gtk::Window& wi
       protocol(std::move(protocol)), name(name),
       screen(window, *this, this->weapons),
       turn(*this, this->screen.getTurnLabel()),
       view list (this->screen.getWorld(), *this, this->screen.getPlayersView(), mus
   icPlayer),
       data receiver (*this).
       handlers(*this, this->view_list, this->weapons, this->screen.getWorld()){
10
11
12
       this->musicPlayer.playMusic();
13
       this->data receiver.start();
14
15
   Player::~Player() {
16
       this->data receiver.stop();
17
18
       this->data_receiver.join();
19
   void Player::startTurn(int worm id, int player id, float wind) {
21
       this->view list.setCurrentWorm(worm id);
22
23
       this->screen.getWindView().update(wind):
       const std::string& current player = this->screen.getPlayersView().getPlayer(
   player id);
       if (current player == this->name) {
25
            //Es mi turno
26
            this->musicPlayer.playStartTurnSound();
27
            this->handlers.enableAll();
28
            this->changeWeapon(this->weapons.getCurrentWeapon().getName());
29
            this->screen.getTurnLabel().beginTurn();
30
            this->turn.start();
31
32
33
            this->screen.getTurnLabel().beginTurn(current_player);
34
35
36
   void Player::endTurn() {
37
       this->screen.getTurnLabel().endTurn();
38
       this->handlers.disableAll();
       this->view list.removeScopeVisibility();
40
       this->protocol.sendEndTurn();
41
42
43
   void Player::endGame(const std::string& winner) {
       this->data_receiver.stop();
45
       this->handlers.disableAll();
46
       this->screen.getTurnLabel().setWinner(winner, this->name == winner);
47
       this->view list.setVictorv();
48
49
50
   void Player::endTurnEarly() {
51
52
       this->turn.stop();
53
54
   void Player::shootWeapon() {
55
       this->turn.reduceTime();
56
       this->weapons.getCurrentWeapon().shoot();
57
58
60
   void Player::changeWeapon(std::string weapon) {
       this->musicPlayer.playSelectWeaponSound();
61
62
       this->weapons.changeWeapon(weapon);
       if (this->handlers.isEnabled()) {
```

```
Plaver.cpp
Jun 06. 18 20:08
                                                                              Page 2/2
            this->protocol.sendChangeWeapon(weapon);
65
66
   void Player::shoot(Position position) {
        this->shootWeapon();
        this->protocol.sendWeaponSelfDirectedShoot(position);
70
        this->screen.getWeaponsView().updateAmmo(this->weapons.getCurrentWeapon());
   void Player::playTickTime() {
        this->musicPlayer.playTickSound();
76
   void Player::shoot(int angle, int power, int time) {
79
        this->shootWeapon();
80
        if (!this->weapons.getCurrentWeapon().isTimed()) {
            time = -1:
81
82
83
        if (!this->weapons.getCurrentWeapon().hasScope())
            angle = MAX WEAPON ANGLE * 8;
        this->protocol.sendWeaponShoot(angle, power, time);
        this->view list.removeScopeVisibility();
        this->screen.getWeaponsView().updateAmmo(this->weapons.getCurrentWeapon());
   ViewsList& Player::getViewsList() {
        return this->view list;
92
93
   ScreenView& Player::getScreen() {
        return this->screen;
97
   WeaponList& Player::getWeapons() {
99
        return this->weapons;
100
101
102
   ClientProtocol& Player::getProtocol() {
103
        return this->protocol;
104
105
107
   MusicPlayer& Player::getMusicPlayer() {
108
        return this->musicPlayer;
109
```

```
Plaver.h
May 31, 18 12:08
                                                                               Page 1/2
    #ifndef __CLIENTPLAYER_H__
2 #define __CLIENTPLAYER_H__
    #include <memory>
   #include <atkmm/window.h>
   #include "ClientProtocol.h"
    #include "Turn.h"
   #include "Weapon.h"
   #include "WeaponList.h"
   #include "ScreenView.h"
#include "ViewsList.h"
12 #include "Position.h"
13 #include "DataReceiver.h"
   #include "Handlers.h"
15
   #include "MusicPlayer.h"
17
   class Player {
        private:
18
            ClientProtocol protocol;
19
20
            std::string name;
21
            WeaponList weapons;
22
            ScreenView screen;
            Turn turn;
23
            ViewsList view list;
24
25
            DataReceiver data receiver;
            Handlers handlers;
26
            MusicPlayer musicPlayer;
27
28
            /* Reduce el tiempo del turno y actualiza la municion */
29
            void shootWeapon();
30
31
32
        public:
            /* Constructor */
33
            Player (ClientProtocol protocol, const std::string& name, Gtk::Window& wi
34
   ndow);
35
            /* Destructor */
36
            ~Player();
37
38
39
            /* Comienza el turno. Si es el turno del jugador entonces,
40
               habilita los handlers, sino muestra los movimientos realizados
               por el otro jugador */
42
            void startTurn(int worm_id, int player_id, float wind);
43
44
            /* Finaliza el turno del jugador actual */
45
46
            void endTurn();
47
            /* Finaliza el juego */
48
            void endGame(const std::string& winner);
49
50
            /* El jugador debe terminar su turno antes */
51
            void endTurnEarly();
52
53
            /* Cambia el arma actual por la espeificada */
54
55
            void changeWeapon(std::string weapon);
56
            /* Realiza el disparo del arma con el angulo, potencia
57
               y tiempo pasados */
58
            void shoot(int angle, int power, int time);
59
60
            /* Realiza el disparo del arma en la posicion pasada */
61
62
            void shoot (Position position);
63
            /* Reproduce el sonido de falta de tiempo */
64
            void playTickTime();
```

```
[75.42] Taller de programacion
                                          Player.h
May 31, 18 12:08
                                                                               Page 2/2
            /* Devuelve la lista de los elementos presentes en la vista */
67
            ViewsList& getViewsList();
68
69
            /* Devuelve la vista */
70
            ScreenView& getScreen();
71
72
            /* Devuelve la lista de armas */
73
            WeaponList& getWeapons();
74
75
            /* Devuelve el protocolo */
77
            ClientProtocol& getProtocol();
78
            /* Devuelve el music player */
79
80
            MusicPlayer& getMusicPlayer();
81
   };
83 #endif
```

```
Scope.cpp
Jun 05. 18 14:07
                                                                              Page 1/1
   #include "Scope.h"
2 #include "Path.h"
   #include "WeaponNames.h"
   Scope::Scope(WorldView& world): world(world) {
5
       this->scope.set(SCOPE IMAGE);
       this->angle = DEFAULT ANGLE;
       this->world.addElement(this->scope, Position(0,0), 0, 0);
8
9
10
   Scope::~Scope(){}
11
12
13
   void Scope::update(int angle, WormView& worm) {
       this->angle = angle;
14
15
       char dir = worm.getDir();
16
       if (dir == DIR_LEFT)
           angle = 180 - \text{angle};
17
       this->world.moveScope(this->scope, worm.getWidget(), angle);
18
       this->scope.show();
19
20
       worm.updateScope(this->angle);
21
22
   void Scope::update(WormView& worm) {
23
       this->update(this->angle, worm);
24
25
26
27
   void Scope::hide(){
28
       if (this->scope.is_visible()){
29
           this->scope.hide();
30
31
32 }
```

```
Scope.h
Jun 02. 18 18:22
                                                                               Page 1/1
    #ifndef ___SCOPE_H__
   #define ___SCOPE_H__
   #include <gtkmm/image.h>
   #include "WorldView.h"
   #include "WormView.h"
   class Scope{
       private:
10
            Gtk:: Image scope;
            WorldView& world;
12
            int angle;
13
14
        public:
15
            /* Constructor */
16
            Scope (WorldView& world);
17
            /* Destructor */
18
19
            ~Scope();
20
21
            /* Actualiza la posicion del scope */
22
            void update(int angle, WormView& worm);
23
24
            /* Actualiza la posicion del scope */
25
            void update (WormView& worm);
26
            /* Esconde el scope */
27
            void hide();
28
29
   };
30
32 #endif
```

```
ViewsList.cpp
Jun 06. 18 20:08
                                                                             Page 1/3
   #include "ViewsList.h"
#include <qlibmm/main.h>
   #include "ObjectSizes.h"
   #include "WeaponNames.h"
   #include "Player.h"
   ViewsList::ViewsList(WorldView& world, Player& player, PlayersList& players list
     MusicPlayer& musicPlayer):
       world(world), player(player), players_list(players_list), scope(world), musi
   cPlayer(musicPlayer) {
10
       this->current_worm_id = -1;
       this->weapon_focused = -1;
11
       this->worm_focused = -1;
12
13
15
   ViewsList::~ViewsList(){}
16
17
18
   void ViewsList::removeWorm(int id){
       auto it = this->worms.find(id):
       if (it != this->worms.end()) {
20
            this->players list.reducePlayerLife(it->second.getPlayerId(), it->second
21
    .aetLife());
22
            if (id == this->current worm id) {
                this->player.endTurnEarly();
23
24
            it->second.removeFromWorld();
25
            this->worms.erase(it):
26
27
            this->musicPlayer.playDeathSound();
            this->checkMovingWorms();
28
29
30
31
   void ViewsList::removeWeapon(int id) {
32
       auto it = this->weapons.find(id);
33
       if (it != this->weapons.end()) {
34
            if(it->second.getName() != BAT NAME) {
35
                this->musicPlayer.playExplosionSound(it->second.getName());
36
                ExplosionView explosion(std::move(it->second));
37
                this->animation.addAndStart(std::move(explosion));
38
39
            this->weapons.erase(it);
40
41
42
            if (this->weapon focused == id) {
                this->weapon focused = -2;
43
                this->checkMovingWorms();
44
45
46
47
   void ViewsList::updateWormData(int id, int player_id, float pos_x, float pos_y,
   int life, char dir, bool colliding) {
       auto it = this->worms.find(id);
50
       Position pos(pos x / UNIT TO SEND, pos y / UNIT TO SEND);
51
52
       if (it == this->worms.end()){
53
            //Worm no existe
            WormView worm(this->world, life, dir, pos, player_id);
54
            this->worms.insert(std::make_pair(id, std::move(worm)));
55
            this->players_list.addPlayerLife(player_id, life);
56
57
            //Worm existe
58
            int current life = it->second.getLife();
59
            if (current life != life) {
60
61
                this->players_list.reducePlayerLife(player_id, current_life - life);
                if (id == this->current worm id) {
```

```
ViewsList.cpp
Jun 06, 18 20:08
                                                                               Page 2/3
                     this->musicPlayer.playDamageReceiveSound();
                     this->player.endTurnEarly();
64
65
66
            it->second.updateData(life, dir, pos, colliding, id == this->current wor
    m id, this->weapon focused != -1);
            this->checkMovingWorms();
69
70
   void ViewsList::updateWeaponData(int id, const std::string& weapon name, float p
   os_x, float pos_y) {
        auto it = this->weapons.find(id);
        Position pos(pos_x / UNIT_TO_SEND, pos_y / UNIT_TO_SEND);
74
75
        if (it == this->weapons.end()){
76
            //Weapon no existe
            BulletView weapon(this->world, weapon_name, pos);
77
78
            if (this->weapon_focused < 0) {</pre>
                weapon.setFocus(true);
79
80
                this->weapon focused = id;
                this->removeWormFocus();
82
            this->weapons.insert(std::make pair(id, std::move(weapon)));
        } else {
85
            //Weapon existe
            it->second.updateData(pos);
86
87
88
89
90
   void ViewsList::changeWeapon(const std::string& weapon name) {
        auto it = this->worms.find(this->current worm id);
        it->second.changeWeapon(weapon_name);
92
        if (WeaponsFactory().createWeapon(weapon_name, 1)->hasScope()) {
93
            this->scope.update(it->second);
94
95
96
97
   void ViewsList::updateScope(int angle) {
        auto it = this->worms.find(this->current worm id);
99
        if (it == this->worms.end()) {
100
101
            return;
102
103
        this->scope.update(angle, it->second);
104
105
   void ViewsList::removeScopeVisibility() {
106
        this->scope.hide();
107
108
109
   bool ViewsList::addGirderCallBack(size t size, Position pos, int rotation) {
        GirderView girder(this->world, size, pos, rotation);
        this->girders.push_back(std::move(girder));
        return false:
113
114
115
    void ViewsList::addGirder(size t size, float pos x, float pos y, int rotation) {
        sigc::slot<br/>bool> my slot = sigc::bind(sigc::mem fun(*this, &ViewsList::addGi
    rderCallBack), size, Position(pos_x, pos_y), rotation);
        Glib::signal_idle().connect(my_slot);
118
119
120
   void ViewsList::setCurrentWorm(int id) {
121
        this->removeWormFocus();
122
        for (auto it = this->worms.begin(); it != this->worms.end(); ++it) {
123
124
            it->second.resetFocus();
125
```

```
ViewsList.cpp
Jun 06. 18 20:08
                                                                                Page 3/3
        this->current_worm_id = id;
        this->worm focused = id;
127
        this->weapon focused = -1;
128
        WormView& worm = this->worms.at(id):
120
        this->world.setFocus(worm.getWidget()):
130
131
        worm.setFocus(true);
132
133
   void ViewsList::removeWormFocus(){
13/
135
        auto it = this->worms.find(this->worm focused);
        if (it != this->worms.end()){
137
            it->second.resetFocus();
138
139
        this->worm_focused = -1;
140
141
142
    void ViewsList::checkMovingWorms() {
        if (this->weapon_focused != -2) {
143
144
145
1/16
147
        auto it = this->worms.find(this->worm focused);
        if (it == this->worms.end() || !it->second.isMoving()){
1/18
            this->removeWormFocus();
1/10
150
            for (auto it2 = this->worms.begin(); it2 != this->worms.end(); ++it2) {
                if (it2->second.isMoving()) {
151
                     this->worm focused = it2->first;
152
                     it2->second.setFocus(true);
153
                     this->world.setFocus(it2->second.getWidget());
154
                     return;
155
156
157
158
159
160
161
    void ViewsList::setVictorv() {
162
        if (this->worms.empty()) {
163
            return:
164
165
        for (auto iter = this->worms.begin(); iter != this->worms.end(); iter++) {
166
167
            this->musicPlayer.playVictory();
            iter->second.setVictory();
168
            this->world.setFocus(iter->second.getWidget());
160
170
171
172
    void ViewsList::shoot(const std::string& weapon) {
173
        this->worms.at(this->current_worm_id).weaponShoot(weapon);
174
175 }
```

```
ViewsList.h
Jun 06. 18 20:08
                                                                              Page 1/2
    #ifndef ___VIEWSLIST_H__
   #define ___VIEWSLIST_H__
   #include <unordered map>
   #include <vector>
   #include <string>
   #include "WorldView.h"
   #include "WormView.h"
   #include "BulletView.h"
   #include "GirderView.h"
   #include "PlayersList.h"
   #include "ExplosionView.h"
   #include "ExplosionViewList.h"
   #include "MusicPlayer.h"
   #include "Scope.h"
   /* Clase que se encarga de almacenar los objetos visibles */
   class ViewsList{
       private:
            WorldView& world:
20
21
            Player& player;
            PlayersList& players list;
            std::unordered_map<int, WormView> worms;
23
            std::unordered_map<int, BulletView> weapons;
24
25
            std::vector<GirderView> girders:
            int current worm id;
            int weapon_focused;
27
            int worm focused;
28
            ExplosionViewList animation;
29
30
            Scope scope:
            MusicPlayer& musicPlayer;
32
            /* Elimina el focus sobre el worm */
33
            void removeWormFocus();
34
35
            /* CallBacks */
36
            bool addGirderCallBack(size_t size, Position pos, int rotation);
37
38
        public:
            /* Constructor */
            ViewsList (WorldView& world, Player& player, PlayersList& players list, M
   usicPlayer& musicPlayer);
            /* Destructor */
43
44
            ~ViewsList();
45
            /* Elimina al worm de la vista actualizando la vida del player */
            void removeWorm(int id);
            /* Elimina la vista del arma y la reemplaza por la animacion de la explo
   sion */
            void removeWeapon(int id);
51
            /* Actualiza la posicion y la vida del worm */
52
            void updateWormData(int id, int player id, float pos x, float pos y, int
     life, char dir, bool colliding);
            /* Actualiza la posicion del arma */
55
            void updateWeaponData(int id, const std::string& weapon_name, float pos_
56
   x, float pos_y);
            /* CallBack de changeWeapon */
            bool changeWeaponCallBack(const std::string &weapon_name);
59
60
61
            /* Actualiza la vista del worm con el arma nueva */
            void changeWeapon(const std::string &weapon name);
```

```
ViewsList.h
Jun 06. 18 20:08
                                                                             Page 2/2
            /* Actualiza la posicion del scope */
64
            void updateScope(int angle);
65
66
            /* Esconde la vista del scope */
67
            void removeScopeVisibility();
68
69
70
            /* Agrega una viga a la vista en la posicion indicada v
             * con la rotacion indicada */
71
72
            void addGirder(size_t size, float pos_x, float pos_y, int rotation);
73
74
            /* Actualiza el worm actual y hace focus en este */
75
            void setCurrentWorm(int id);
76
77
            /* Actualiza la imagen de los worms ganadores por la animacion
78
             * de los worms festejando */
79
            void setVictory();
80
            /* Chequea si el qusano actual se esta moviendo, caso contario
81
82
            le da el focus a otro */
83
            void checkMovingWorms();
            /* Realiza la animacion del disparo del arma */
85
            void shoot(const std::string& weapon);
87
   };
   #endif
```

```
WeaponPowerAccum.cpp
Jun 05. 18 14:07
                                                                             Page 1/1
   #include "WeaponPowerAccum.h"
   #include "Handlers.h"
   const int TIME STEP = 50;
   const int MINIMUM POWER = 1000;
   const int POWER STEP = 15;
   WeaponPowerAccum:: WeaponPowerAccum (Handlers& handlers, int time) :
        actual time(0), max time(time), handlers(handlers) {}
   WeaponPowerAccum::~WeaponPowerAccum() {}
   bool WeaponPowerAccum::startCallBack() {
        this->actual_time += TIME_STEP;
15
        this->power += POWER_STEP;
16
17
        if (this->actual_time == this->max_time) {
           this->handlers.powerAccumStopped(this->power);
18
19
            return false;
20
21
        return true;
22
   void WeaponPowerAccum::start() {
24
25
        this->actual time = 0;
        this->power = MINIMUM POWER;
26
        this->my_connection = Glib::signal_timeout().connect(sigc::mem_fun(*this, &W
   eaponPowerAccum::startCallBack), TIME STEP);
28
29
   void WeaponPowerAccum::stop() {
30
        if (this->my_connection.connected()) {
           this->my_connection.disconnect();
32
            this->handlers.powerAccumStopped(this->power);
33
34
35
```

```
WeaponPowerAccum.h
May 31, 18 12:08
                                                                             Page 1/1
   #ifndef __CLIENTTIMER_H__
   #define CLIENTTIMER H
   #include <qlibmm/main.h>
   class Handlers;
    /* Clase que simula a un contador */
   class WeaponPowerAccum {
10
       private:
            int actual time;
12
            int max_time;
13
            int power;
            Handlers& handlers;
14
15
            sigc::connection my_connection;
16
17
            /* Callback de start */
           bool startCallBack();
18
19
20
        public:
21
            /* Constructor */
22
            WeaponPowerAccum (Handlers& handlers, int time);
23
            /* Destructor */
24
25
            ~WeaponPowerAccum();
26
            /* Cuenta el tiempo transcurrido y llama al metodo timerStopped
27
               de la clase Handler con este tiempo */
28
            void start();
29
30
            /* Detiene el contador */
31
32
            void stop();
33
   };
   #endif
```

```
ClientProtocol.cpp
Jun 06. 18 20:16
                                                                              Page 1/2
    #include "ClientProtocol.h"
   #include <string>
   #include "Player.h"
   #include "WeaponList.h"
   #include "ObjectSizes.h"
   #include "ServerFatalError.h"
   ClientProtocol::ClientProtocol(Socket&& socket, Gtk::Window& window): Protocol(s
   td::move(socket)), window(window){}
   ClientProtocol::ClientProtocol(ClientProtocol&& other): Protocol(std::move(other
   )), window(other.window) {}
12
   ClientProtocol::~ClientProtocol(){}
   void ClientProtocol::sendMoveAction(char action){
        Buffer buffer:
       buffer.setNext(ACTION);
16
17
        buffer.setNext(MOVE_ACTION);
18
        buffer.setNext(action);
19
        this->sendBuffer(buffer);
20
21
22
   void ClientProtocol::sendChangeWeapon(const std::string &weapon) {
23
        Buffer buffer:
        buffer.setNext(ACTION);
24
        buffer.setNext(CHANGE_WEAPON_ACTION);
25
        this->sendStringBuffer(buffer, weapon);
26
27
        this->sendBuffer(buffer);
28
29
   void ClientProtocol::sendWeaponShoot(int32_t angle, int32_t power, int32_t time)
        Buffer buffer;
31
        buffer.setNext(ACTION);
32
33
        buffer.setNext(SHOOT_WEAPON);
34
        this->sendIntBuffer(buffer, angle);
        this->sendIntBuffer(buffer, power);
35
        this->sendIntBuffer(buffer, time);
36
        this->sendBuffer(buffer);
37
38
   void ClientProtocol::sendWeaponSelfDirectedShoot(const Position &pos) {
        Buffer buffer;
41
42
        buffer.setNext(ACTION);
        buffer.setNext(SHOOT_SELF_DIRECTED);
43
45
        this->sendIntBuffer(buffer, pos.getX() * UNIT_TO_SEND);
        this->sendIntBuffer(buffer, pos.getY() * UNIT_TO_SEND);
46
47
        this->sendBuffer(buffer):
48
49
50
51  void ClientProtocol::updateScope(int angle) {
        Buffer buffer;
52
53
        buffer.setNext(ACTION);
54
        buffer.setNext(MOVE SCOPE);
55
        this->sendIntBuffer(buffer, angle);
56
57
        this->sendBuffer(buffer);
58
59
   void ClientProtocol::sendEndTurn() {
61
        Buffer buffer:
62
        buffer.setNext(END_TURN);
```

```
ClientProtocol.cpp
Jun 06. 18 20:16
                                                                              Page 2/2
        this->sendBuffer(buffer);
65
66
    void ClientProtocol::receiveStartGame() {
67
        Buffer buffer = std::move(this->receiveBuffer()):
68
60
70
   void ClientProtocol::receiveBackgroundImage(WorldView& world) {
71
        Buffer buffer = std::move(this->receiveBuffer()):
72
73
        world.setBackgroundImage(buffer);
74
75
76
   void ClientProtocol::receivePlayers(PlayersList& players_list) {
        Buffer buffer = std::move(this->receiveBuffer());
77
78
        int quantity = this->receiveIntBuffer(buffer);
79
80
        for (int i = 0; i < quantity; i++) {
            Buffer buffer = std::move(this->receiveBuffer());
81
82
83
            int id = this->receiveIntBuffer(buffer);
84
            std::string name = this->receiveStringBuffer(buffer);
85
            players list.addPlayer(id, name);
86
87
88
89
    void ClientProtocol::receiveGirders(ViewsList& viewsList){
90
        Buffer buffer = std::move(this->receiveBuffer());
91
        int quantity = this->receiveIntBuffer(buffer);
92
93
        for (int i = 0; i < quantity; i++){</pre>
94
            Buffer buffer = std::move(this->receiveBuffer());;
95
96
            int size = this->receiveIntBuffer(buffer);
97
            float pos_x = this->receiveIntBuffer(buffer) / UNIT_TO_SEND;
98
            float pos_y = this->receiveIntBuffer(buffer) / UNIT_TO_SEND;
99
            int rotation = this->receiveIntBuffer(buffer);
100
            viewsList.addGirder(size, pos_x, pos_y, rotation);
101
102
103
104
    void ClientProtocol::receiveWeaponsAmmo(WeaponList& weapon list){
        Buffer buffer = std::move(this->receiveBuffer());
106
        int quantity = this->receiveIntBuffer(buffer);
107
108
        for (int i = 0; i < quantity; i++) {
109
            Buffer buffer = std::move(this->receiveBuffer());
110
111
            std::string name = this->receiveStringBuffer(buffer);
112
            int ammo = this->receiveIntBuffer(buffer);
113
            weapon list.add(name, ammo);
114
115
116
117
   void ClientProtocol::sendBuffer(Buffer &buffer) {
118
119
            Protocol::sendBuffer(buffer);
120
         catch (const std::exception& e) {
121
            ServerFatalError error (this->window);
122
123
124 }
```

```
May 31, 18 16:52
                                    ClientProtocol.h
                                                                             Page 1/2
   #ifndef __CLIENTPROTOCOL_H__
   #define CLIENTPROTOCOL H
   #include "Socket.h"
   #include "Protocol.h"
   #include "Position.h"
   #include "ViewsList.h"
   #include "PlayersList.h"
   #include <gtkmm/window.h>
11 class Player;
12 class WeaponList;
   /* Clase que se encarga de enviar y recibir mensajes del socket
    * con un formato determinado */
   class ClientProtocol: public Protocol {
       private:
18
            Gtk::Window& window;
19
20
21
            /* Constructor */
22
            ClientProtocol(Socket&& socket, Gtk::Window& window);
23
24
            /* Constructor por movimiento */
25
            ClientProtocol (ClientProtocol&& other):
            /* Destructor */
27
            ~ClientProtocol():
28
29
30
            /* Envia un mensaje que indica una accion de movimiento */
            void sendMoveAction(char action);
32
            /* Envia un mensaje que indica una accion de cambio de arma
33
34
             * con el nombre del arma */
            void sendChangeWeapon(const std::string &weapon);
35
36
            /* Envia un mensaje de accion de disparo, con el angulo, la potencia
37
             * v el tiempo de explosion */
38
            void sendWeaponShoot(int32_t angle, int32_t power, int32_t time);
39
40
            /* Envia un mensaje de accion de disparo teledirigido con
             * la posicion del disparo */
42
            void sendWeaponSelfDirectedShoot(const Position &pos);
43
44
            /* Envia un mesaje que indica el cambio del angulo del scope */
45
            void updateScope(int angle);
46
47
            /* Envia un mensaje de finalizacion de turno */
48
            void sendEndTurn();
49
50
            /* Recibe el comienzo del juego */
            void receiveStartGame();
52
53
54
            /* Recibe y setea la imagen de fondo */
            void receiveBackgroundImage(WorldView& world);
55
56
57
            /* Recibe los jugadores de la partida junto con su
             * id v su nombre */
58
            void receivePlayers(PlayersList& players list);
59
60
            /* Recibe la vigas presentes en el mapa junto con su tamaño,
61
             * su posicion y su rotacion */
62
            void receiveGirders(ViewsList& viewsList);
63
            /* Recibe las armas presentes en el juego junto con
65
             * su municion */
```

```
May 31, 18 16:52

ClientProtocol.h

Page 2/2

roid receiveWeaponsAmmo(WeaponList& weapon_list);

/* Envia el contenido del buffer */
ro void sendBuffer(Buffer &buffer) override;

ri };

rz

rs #endif
```

```
DataReceiver.cpp
Jun 06. 18 20:08
                                                                             Page 1/2
    #include "DataReceiver.h"
   #include "Player.h"
   #include <glibmm/main.h>
   #include "ObjectSizes.h"
   DataReceiver::DataReceiver(Player& player):
        player(player), protocol(player.getProtocol()) {}
   DataReceiver::~DataReceiver() {
        this->protocol.stop();
   void DataReceiver::run(){
15
            this->initialConfig();
16
            while (this->running) {
                Buffer data = this->protocol.receiveBuffer();
17
                sigc::slot<bool> my_slot = sigc::bind(sigc::mem_fun(*this, &DataRece
   iver::analizeReceivedData), data);
                Glib::signal_idle().connect(my_slot);
20
21
        } catch (const std::exception& e) {
22
            if (this->running) {
23
24
                this->player.getScreen().close();
25
26
27
28
   void DataReceiver::initialConfig() {
29
        this->protocol.receiveStartGame();
30
        this->protocol.receiveBackgroundImage(this->player.getScreen().getWorld());
        this->protocol.receivePlayers(this->player.getScreen().getPlayersView());
32
        this->protocol.receiveGirders(this->player.getViewsList());
33
        this->protocol.receiveWeaponsAmmo(this->player.getWeapons());
34
35
        this->player.getScreen().show();
36
37
   bool DataReceiver::analizeReceivedData(Buffer buffer) {
38
        char action = buffer.getNext();
40
        if (action == START TURN) {
            int worm id = Protocol::receiveIntBuffer(buffer);
42
            int player_id = Protocol::receiveIntBuffer(buffer);
43
44
            float wind = Protocol::receiveIntBuffer(buffer) / UNIT TO SEND;
            this->player.startTurn(worm_id, player_id, wind);
45
46
        } else if (action == END_GAME) {
            std::string winner = Protocol::receiveStringBuffer(buffer);
47
            this->player.endGame(winner);
48
        } else if (action == END TURN) {
49
            this->player.endTurnEarly();
50
        } else if (action == CHANGE_WEAPON_ACTION)
            std::string weapon(Protocol::receiveStringBuffer(buffer));
52
53
            this->player.getViewsList().removeScopeVisibility();
            this->player.getViewsList().changeWeapon(weapon);
54
55
        } else if (action == MOVE SCOPE)
56
            int angle = Protocol::receiveIntBuffer(buffer);
            this->player.getViewsList().updateScope(angle);
57
58
        } else if (action == SHOOT_WEAPON_ACTION) {
            std::string weapon(Protocol::receiveStringBuffer(buffer));
59
            this->player.getViewsList().removeScopeVisibility();
60
            this->player.getViewsList().shoot(weapon);
61
            this->player.getMusicPlayer().playWeaponShotSound(weapon);
62
        } else if (action == MOVING_OBJECT) {
63
            char type = buffer.getNext();
64
            int id = Protocol::receiveIntBuffer(buffer);
```

```
DataReceiver.cpp
Jun 06. 18 20:08
                                                                             Page 2/2
           if (type == WORM_TYPE) {
67
                int player id = Protocol::receiveIntBuffer(buffer);
68
                int pos_x = Protocol::receiveIntBuffer(buffer);
60
                int pos_y = Protocol::receiveIntBuffer(buffer);
70
                int life = Protocol::receiveIntBuffer(buffer);
71
72
                char dir = buffer.getNext();
                bool colliding = buffer.getNext();
73
                this->player.getViewsList().updateWormData(id, player_id, pos_x, pos
7/
   _y, life, dir, colliding);
                this->player.getViewsList().removeScopeVisibility();
76
             else if (type == WEAPON_TYPE) {
77
                std::string weapon(Protocol::receiveStringBuffer(buffer));
78
79
                int pos x = Protocol::receiveIntBuffer(buffer);
80
                int pos_y = Protocol::receiveIntBuffer(buffer);
81
                this->player.getViewsList().updateWeaponData(id, weapon, pos_x, pos_
   у);
82
83
       } else if (action == DEAD_OBJECT) {
84
            char type = buffer.getNext();
            int id = Protocol::receiveIntBuffer(buffer);
           if (type == WORM_TYPE) {
86
                this->player.getViewsList().removeWorm(id);
87
88
             else if (type == WEAPON_TYPE) {
                this->player.getViewsList().removeWeapon(id);
89
90
       } else if (action == MOVE ACTION) {
91
            char movement = buffer.getNext();
92
           this->player.getMusicPlayer().playJumpSound(movement);
93
       return false;
96 }
```

```
DataReceiver.h
Jun 06. 18 20:08
                                                                              Page 1/1
   #ifndef __DATARECEIVER_H__
   #define __DATARECEIVER_H__
   #include "Thread.h"
   #include "ClientProtocol.h"
   class Player;
   /* Clase que se encarga de recibir los mensajes enviados por el servidor */
   class DataReceiver: public Thread{
        private:
            Player& player;
13
            ClientProtocol& protocol;
15
            /* Recibe los datos de la configuracion inicial */
16
            void initialConfig();
            /* Analiza los datos recibidos */
18
19
            bool analizeReceivedData(Buffer buffer);
20
21
       public:
            /* Constructor */
            DataReceiver (Player& player);
23
24
25
            /* Destructor */
26
            ~DataReceiver();
27
            /* Comienza a recibir mensajes del protocolo */
28
            void run() override;
29
   };
30
   #endif
33
```

```
Handlers.cpp
Jun 06. 18 20:08
                                                                             Page 1/3
   #include "Handlers.h"
#include <qtkmm/adjustment.h>
   #include <qdk/qdkkeysyms.h>
   #include "Player.h"
   #include "ViewPositionTransformer.h"
   #include "WeaponNames.h"
8
   const char SPACE = '';
   const int WEAPONS DEFAULT TIME = 3;
10 const char ASCII OFFSET = 48;
11 const char ASCII 1 = 49;
   const char ASCII_5 = 53;
   const int MAX_TIME = 3000;
   const int ANGLE_STEP = 6;
15
   Handlers::Handlers(Player& player, ViewsList& view list, WeaponList& weapons, Wo
   rldView& world):
       player(player), view_list(view_list), weapons(weapons), world(world),
17
       scroll_handler(world.getWindow()), power_accumulator(*this, MAX_TIME) {
18
19
            this->has shoot = false:
20
            this->current angle = DEFAULT ANGLE;
           this->weapons_time = WEAPONS_DEFAULT_TIME;
21
            this->enabled = false;
22
23
24
   Handlers::~Handlers() {}
25
26
   void Handlers::enableAll() {
27
       this->weapons time = WEAPONS DEFAULT TIME:
28
       this->current angle = DEFAULT ANGLE;
29
       this->has shoot = false:
30
31
       this->player.getProtocol().updateScope(DEFAULT_ANGLE);
32
33
34
       this->world.getWindow().get_parent()->get_parent()->set_can_focus(true);
35
       this->world.getWindow().get_parent()->get_parent()->grab_focus();
36
       this->world.getWindow().get_parent()->get_parent()->signal_key_press_event()
37
    .connect(sigc::mem_fun(*this,
38
                       &Handlers::keyPressHandler));
       this->world.getWindow().get parent()->get parent()->signal key release event
    ().connect(sigc::mem fun(*this,
40
                         &Handlers::keyReleaseHandler));
       this->world.qetWindow().signal_button_press_event().connect(sigc::mem_fun(*t
41
   his, &Handlers::onButtonPressEvent));
42
       this->enabled = true;
43
44
45
   void Handlers::disableAll() {
       this->world.qetWindow().qet_parent()->qet_parent()->signal_key_press_event()
47
    .connect(sigc::mem_fun(*this,
48
                       &Handlers::inactiveKevHandler));
       this->world.getWindow().get parent()->get parent()->signal key release event
49
     .connect(sigc::mem_fun(*this,
50
                         &Handlers::inactiveKevHandler));
       this->world.getWindow().signal_button_press_event().connect(sigc::mem_fun(*t
51
   his, &Handlers::inactiveButtonHandler));
52
       this->enabled = false:
53
54
```

```
Handlers.cpp
Jun 06. 18 20:08
                                                                              Page 2/3
    void Handlers::powerAccumStopped(int power) {
        this->player.shoot(this->current_angle, power, this->weapons_time);
58
59
   bool Handlers::kevPressHandler(GdkEventKey *key event) {
60
        if (key event->keyval == GDK KEY Left) {
61
            this->player.getProtocol().sendMoveAction(MOVE LEFT);
62
        } else if (key event->keyval == GDK KEY Right) {
63
            this->player.getProtocol().sendMoveAction(MOVE RIGHT);
64
65
        } else if (key event->keyval == GDK KEY Return) {
            this->player.getProtocol().sendMoveAction(JUMP);
        } else if (key_event->keyval == GDK_KEY_BackSpace)
            this->player.getProtocol().sendMoveAction(ROLLBACK);
68
69
        } else if (key_event->keyval == GDK_KEY_Up) {
70
            if (!this->weapons.getCurrentWeapon().hasScope()) {
71
                return true;
72
            if (this->current_angle < MAX_WEAPON_ANGLE) {</pre>
73
74
                this->current angle += ANGLE STEP;
75
76
            this->player.getProtocol().updateScope(this->current angle);
77
         else if (key event->keyval == GDK KEY Down) {
            if (!this->weapons.getCurrentWeapon().hasScope()) {
78
79
                return true:
80
            if (this->current angle > MIN WEAPON ANGLE) {
81
                this->current angle -= ANGLE STEP;
82
83
84
            this->player.getProtocol().updateScope(this->current_angle);
85
         else if (key event->keyval >= ASCII 1 && key event->keyval <= ASCII 5) {
            this->weapons_time = key_event->keyval - ASCII_OFFSET;
86
        } else if (key_event->keyval == SPACE && key_event->type == GDK_KEY_PRESS)
            if (this->weapons.getCurrentWeapon().isSelfDirected()) {
88
                return true:
89
90
91
            if (!this->weapons.getCurrentWeapon().hasAmmo()) {
92
                return true:
93
            if (this->has shoot) {
94
                return true:
95
96
            this->has shoot = true;
            if (!this->weapons.getCurrentWeapon().hasVariablePower()) {
                this->player.shoot(this->current_angle, -1, this->weapons_time);
99
100
                this->power accumulator.start();
101
102
103
104
        return true:
105
106
   bool Handlers::keyReleaseHandler(GdkEventKey *key_event) {
        if (key_event->type == GDK_KEY_RELEASE) {
108
            if (key_event->keyval == SPACE) {
109
110
                if (this->weapons.getCurrentWeapon().isSelfDirected()) {
111
112
                if (!this->weapons.getCurrentWeapon().hasVariablePower()) {
113
                    return true;
114
115
                if (!this->weapons.getCurrentWeapon().hasAmmo()) {
116
                    this->player.getMusicPlayer().playNoAmmo();
117
118
                    return true:
119
                this->power_accumulator.stop();
120
121
```

```
Handlers.cpp
Jun 06. 18 20:08
                                                                                Page 3/3
123
        return true:
124
125
   bool Handlers::onButtonPressEvent(GdkEventButton *event)
126
127
        if (!this->weapons.getCurrentWeapon().isSelfDirected()) {
128
            return true;
129
        if (!this->weapons.getCurrentWeapon().hasAmmo()) {
130
131
            this->player.getMusicPlayer().playNoAmmo();
132
            return true:
133
134
        if (this->has_shoot) {
            return true:
135
136
137
        if ((event->type == GDK BUTTON PRESS) && (event->button == 1)) {
138
            float x = event->x;
            float y = event->y;
139
            x += this->world.getWindow().get_hadjustment()->get_value();
140
141
            y += this->world.getWindow().get_vadjustment()->get_value();
142
            Position position (x, y);
            Position newPosition = ViewPositionTransformer(this->world.getLayout())
    transformToPosition(position);
            this->has shoot = true;
1//
            this->player.shoot(newPosition);
145
146
147
        return true;
148
149
   bool Handlers::inactiveKeyHandler(GdkEventKey *key_event) {
150
        return true:
151
152
153
   bool Handlers::inactiveButtonHandler(GdkEventButton *event) {
154
        return true:
155
156
157
    int Handlers::getCurrentAngle() const{
158
        return this->current_angle;
159
160
161
   bool Handlers::isEnabled() {
        return this->enabled;
163
164
```

```
Handlers.h
Jun 05. 18 14:07
                                                                              Page 1/2
    #ifndef __HANDLERS__H_
   #define HANDLERS H
   #include <adk/adk.h>
   #include "WeaponPowerAccum.h"
   #include "ScrollHandler.h"
   class Player;
   class ViewsList:
   class WeaponList;
11 class WorldView:
   /* Clase que se encarga de definir los handlers del teclado y
      del mouse. */
15
   class Handlers{
16
       private:
17
            Player& player;
            ViewsList& view list:
18
            WeaponList& weapons:
19
20
            WorldView& world:
21
            ScrollHandler scroll handler;
22
            bool has shoot;
23
24
            int current_angle;
25
            int weapons time;
            bool enabled;
26
27
            WeaponPowerAccum power accumulator;
28
29
30
        public:
31
            /* Constructor */
            Handlers (Player& player, ViewsList& view_list, WeaponList& weapons, Worl
33
   dView& world);
34
            /* Destructor */
35
36
            ~Handlers();
37
            /* Handler completo para el presionado de teclas. Indica
38
               los pasos que se deben realizar al presionar una tecla
39
               especifica */
40
            bool keyPressHandler(GdkEventKey *key event);
42
13
            /* Handler completo para la liberaciÃ3n de teclas. Indica
44
               los pasos que se deben realizar al liberar una tecla
               especifica */
45
            bool keyReleaseHandler (GdkEventKey *key_event);
46
47
            /* Handler del mouse. Indica los pasos que se deben realizar
48
               al utilizar el mouse */
49
            bool onButtonPressEvent(GdkEventButton *event);
50
51
            /* Handler inactivo de las teclas. Indica que no se debe
52
               realizar ninguna accion al producirse un evento */
53
            bool inactiveKeyHandler(GdkEventKey *key event);
54
55
56
            /* Handler inactivo del mouse. Indica que no se debe
               realizar ninguna accion al producirse un evento */
57
            bool inactiveButtonHandler(GdkEventButton *event);
58
59
            /* Habilita todos los handlers */
60
61
            void enableAll();
62
            /* Deshabilita todos los handlers */
63
            void disableAll();
64
```

```
Handlers.h
Jun 05. 18 14:07
                                                                            Page 2/2
            /* Realiza el shoot del player */
           void powerAccumStopped(int power);
67
68
            /* Devuelve el angulo actual del scope */
60
            int getCurrentAngle() const;
70
71
            /* Devuelve true si estan habilitados los handlers */
72
           bool isEnabled();
73
74
   };
   #endif
```

```
PlayerLifeLabel.cpp
May 30, 18 20:03
                                                                                   Page 1/1
    #include "PlayerLifeLabel.h"
   #include "GamePlayers.h"
   const std::string begining("<span color='");</pre>
   const std::string middle("'>");
   const std::string ending("</span>");
   PlayerLifeLabel::PlayerLifeLabel(): id(0), player_name(""), life(0){
        this->label.set use markup(true);
10
   PlayerLifeLabel::~PlayerLifeLabel(){}
   void PlayerLifeLabel::setPlayerName(int id, const std::string& player_name) {
        this -> id = id;
16
        this->player_name = player_name;
17
        this->updateLabel();
18
19
20
   void PlayerLifeLabel::addLife(int life) {
21
        this->life += life;
        this->updateLabel();
23
24
25
   void PlayerLifeLabel::reduceLife(int life) {
        this -> life -= life;
26
        this->updateLabel();
27
28
29
   Gtk::Label& PlayerLifeLabel::getLabel() {
30
        return this->label;
31
void PlayerLifeLabel::updateLabel(){
        std::string message = begining + colors[this->id] + middle;
message += std::to_string(this->id) + "-" + this->player_name;
35
36
        message += ":" + std::to_string(this->life) + ending;
        this->label.set_markup(message);
39 }
```

PlaverLifeLabel.h May 30, 18 20:03 Page 1/1 #ifndef ___PLAYERLIFELABEL_H__ #define ___PLAYERLIFELABEL_H__ #include <qtkmm/label.h> /* Clase que se encarga de controlar el indicador de vida del jugador */ class PlayerLifeLabel{ private: int id; 10 std::string player name; int life; Gtk::Label label; 13 /* Actualiza la informacion del label */ 14 15 void updateLabel(); 16 17 public: /* Constructor */ 18 19 PlayerLifeLabel(); 20 21 /* Destructor */ 22 ~PlayerLifeLabel(); 23 24 25 /* Establece el nombre del jugador */ void setPlayerName(int id, const std::string& player_name); 26 27 /* Agrega la vida al label */ 28 void addLife(int life); 29 30 /* Disminuye la vida y actualiza la vista del label */ 31 void reduceLife(int life); 32 33 /* Devuelve el label del jugador */ 34 Gtk::Label& getLabel(); 35 36 }; 37 38 #endif

```
PlaversList.cpp
Jun 06. 18 20:08
                                                                              Page 1/1
   #include "PlayersList.h"
   #include <qlibmm/main.h>
   #define SPACING 20
   PlayersList::PlayersList(): container(false, SPACING) {
        this->title.set use markup(true);
        this->title.set_markup("<span><b><u>Jugadores</u></b></span>");
        this->container.pack start (this->title, Gtk::PACK SHRINK);
10
   PlayersList::~PlayersList(){}
   void PlayersList::addPlayer(int id, const std::string& name) {
       sigc::slot<bool> my_slot = sigc::bind(sigc::mem_fun(*this, &PlayersList::add
   PLaverCallBack), id, name);
       Glib::signal_idle().connect(my_slot);
17
18
19
   bool PlayersList::addPLayerCallBack(int id, std::string name) {
20
        this->players[id] = name;
        this->labels[id].setPlayerName(id, name);
        this->container.pack_start(this->labels[id].getLabel(), Gtk::PACK_SHRINK);
22
        return false:
23
24
25
   const std::string& PlayersList::getPlayer(int id) const{
        return this->players.at(id);
27
28
29
   Gtk::Container& PlayersList::getWindow() {
30
        return this->container;
32
33
   void PlayersList::addPlayerLife(int player_id, int life){
34
35
        this->labels[player_id].addLife(life);
36
37
   void PlayersList::reducePlayerLife(int player_id, int life) {
38
        this->labels[player_id].reduceLife(life);
39
40
```

```
PlaversList.h
May 30, 18 20:03
                                                                             Page 1/1
   #ifndef ___PLAYERSLIST_H__
2 #define __PLAYERSLIST_H__
   #include <map>
1
   #include <string>
   #include <qtkmm/hvbox.h>
   #include <qtkmm/label.h>
   #include "PlayerLifeLabel.h"
   /* Clase que se encarga de almacenar los nombres y las vidas
    * de todos los jugadores */
   class PlayersList{
13
       private:
14
            std::map<int, std::string> players;
15
            std::map<int. PlayerLifeLabel> labels:
16
            Gtk:: VBox container:
17
            Gtk::Label title:
18
19
            bool addPLayerCallBack(int id, std::string name);
20
21
        public:
            /* Constructor */
22
            PlayersList();
23
24
25
            /* Destructor */
            ~PlayersList();
26
27
            /* Agrega al jugador a la lista de jugadores y agrega su
28
             * informacion a la vista */
29
            void addPlayer(int id, const std::string& name);
30
31
            /* Devuelve el nombre del jugador */
32
            const std::string& getPlayer(int id) const;
33
34
            /* Devuelve el contenedor de los jugadores */
35
36
            Gtk::Container& getWindow();
37
            /* Agrega la informacion de la vida del jugador a la vista */
38
            void addPlayerLife(int player_id, int life);
39
40
            /* Reduce la vida del jugador y actualiza la vista */
41
            void reducePlayerLife(int player id, int life);
42
43
   #endif
```

```
ScreenView.cpp
Jun 06. 18 20:08
                                                                              Page 1/1
    #include "ScreenView.h"
   #include "ServerFatalError.h"
   #include <qlibmm/main.h>
   #define PADDING 10
   #define SPACING 30
   ScreenView::ScreenView(Gtk::Window& window, Player& player, WeaponList& weapons)
        left view(false, SPACING), window(window), weapons view(weapons, player) {
        this->left view.pack start(this->wind view.getWindow(), Gtk::PACK SHRINK);
        this->left_view.pack_start(this->players.getWindow(), Gtk::PACK_SHRINK);
        this->world_box.pack_start(this->left_view, Gtk::PACK_SHRINK, PADDING);
12
        this->world_box.pack_start(this->world.getContainer());
13
14
        this->world box.pack end(this->weapons view.getWindow(), Gtk::PACK SHRINK);
15
16
        this->screen.pack_start(this->turn_label.getWindow(), Gtk::PACK_SHRINK);
17
        this->screen.pack_end(this->world_box);
18
19
20
   ScreenView::~ScreenView() {}
   void ScreenView::show(){
        sigc::slot<bool> my_slot = sigc::mem_fun(*this, &ScreenView::showCallBack);
23
24
        Glib::signal idle().connect(my slot);
25
   bool ScreenView::showCallBack() {
27
        this->weapons_view.update();
28
        this->window.remove();
29
        this->window.add(this->screen);
30
        this->window.show_all();
        return false:
32
33
34
35
   void ScreenView::close(){
        sigc::slot<bool> my_slot = sigc::mem_fun(*this, &ScreenView::closeCallBack);
36
        Glib::signal_idle().connect(my_slot);
37
38
39
   bool ScreenView::closeCallBack() {
        ServerFatalError error (this->window);
        return false;
42
43
44
   WorldView& ScreenView::getWorld() {
46
        return this->world;
47
48
   WeaponView& ScreenView::getWeaponsView() {
49
        return this->weapons view;
50
51
52
   TurnLabel& ScreenView::getTurnLabel() {
53
54
        return this->turn label;
55
56
   PlayersList& ScreenView::getPlayersView() {
57
        return this->players;
58
59
   WindView& ScreenView::getWindView() {
        return this->wind view:
63
```

```
ScreenView.h
May 31, 18 12:08
                                                                              Page 1/1
    #ifndef __CLIENTSCREENVIEW_H__
2 #define CLIENTSCREENVIEW H
    #include <atkmm/hvbox.h>
    #include <qtkmm/label.h>
   #include <atkmm/window.h>
    #include "WorldView.h"
   #include "WeaponView.h"
   #include "TurnLabel.h"
   #include "PlayersList.h"
   #include "WindView.h"
13
    /* Clase que se encarga de almacenar los contenedores principales
    * de la vista y mostrar su contenido */
15
   class ScreenView {
16
        private:
17
            Gtk:: VBox screen;
            Gtk::HBox world_box;
18
19
            Gtk::VBox left view:
20
            Gtk::Window& window;
21
22
            WorldView world;
            WeaponView weapons view;
23
            TurnLabel turn label;
24
25
            PlayersList players;
            WindView wind view;
26
27
            /* CallBacks */
28
            bool showCallBack();
29
            bool closeCallBack();
30
31
32
        public:
            /* Constructor */
33
            ScreenView(Gtk::Window& window, Player& player, WeaponList& weapons);
34
35
            /* Destructor */
36
37
            ~ScreenView();
38
            /* Muestra la pantalla en la ventana */
39
            void show():
40
41
42
            /* Cierra la ventana completamente */
            void close();
43
44
45
            /* Devuelve el WorldView */
            WorldView& getWorld();
46
47
            /* Devuelve el WeaponView */
48
            WeaponView& getWeaponsView();
49
50
            /* Devuelve el TurnLabel */
51
            TurnLabel& getTurnLabel();
52
53
            /* Devuelve el Plavers view */
54
            PlayersList& getPlayersView();
55
56
57
            /* Devuelve el wind view */
            WindView& getWindView();
58
   };
59
   #endif
```

```
TurnLabel.cpp
May 31, 18 12:08
                                                                               Page 1/1
    #include "TurnLabel.h"
   #include <string>
   const std::string begining("<span size='20000'>");
   const std::string ending("</span>");
   TurnLabel::TurnLabel() {
        this->message.set use markup(true);
        this->message.set markup(begining + "Worms" + ending);
10
        this->label.pack start(this->message);
        this->label.pack end(this->time);
12
14
   TurnLabel::~TurnLabel() {}
16
   void TurnLabel::beginTurn()
        std::string message = begining + "Tu turno" + ending;
        this->message.set_markup(message);
18
19
20
   void TurnLabel::beginTurn(const std::string& player name) {
        std::string message = begining + "Turno de" + player name + ending;
        this->message.set markup(message);
23
24
25
   void TurnLabel::endTurn()
26
        this->time.set_markup("");
27
        this->message.set markup (begining + "Termino tu turno" + ending);
28
29
30
   void TurnLabel::setTime(int time) {
        this->time.set_markup(begining + std::to_string(time) + ending);
33
   void TurnLabel::setWinner(const std::string& winner, bool i_win) {
        this->message.set_markup(begining + "Termino el juego" + ending);
36
        std::string winner_message;
37
        if (winner.empty()){
38
            winner_message = "Empate";
39
        } else if (i win) {
40
            winner_message = "GANASTE!!!!";
41
42
            winner message = "Perdiste: (El ganador fue: " + winner;
43
44
45
        this->time.set_markup(begining + winner_message + ending);
46
47
   Gtk::Container& TurnLabel::getWindow() {
        return this->label;
50
```

```
TurnLabel.h
May 31, 18 12:08
                                                                             Page 1/1
   #ifndef ___TURNLABEL_H__
   #define ___TURNLABEL_H__
   #include <qtkmm/hvbox.h>
   #include <qtkmm/label.h>
    /* Clase que se encarga de controlar los labels que indican
    * el estado del turno */
   class TurnLabel{
10
       private:
            Gtk::Label message;
            Gtk::Label time;
13
            Gtk::HBox label;
14
15
        public:
16
            /* Constructor */
17
            TurnLabel():
18
            /* Destructor */
19
20
            ~TurnLabel();
21
22
            /* Cambia el label indicando que es el turno del jugador */
23
            void beginTurn();
24
25
            /* Cambia el label indicando que es el turno del jugador
26
             * con nombre pasado por parametro */
27
            void beginTurn(const std::string& player name);
28
29
            /* Cambia el label indicando que finalizo el turno del jugador */
30
            void endTurn();
31
32
            /* Cambia el label mostrando al ganador */
33
            void setWinner(const std::string& winner, bool i_win);
34
35
            /* Cambia el label de tiempo al tiempo pasado por parametro */
36
37
            void setTime(int time);
38
            /* Devuelve el contenedor de la vista */
39
            Gtk::Container& getWindow();
40
41
   };
   #endif
```

```
WeaponButton.cpp
Jun 05. 18 14:07
                                                                                Page 1/1
    #include "WeaponButton.h"
   #include "Player.h"
   #include "Path.h"
   WeaponButton::WeaponButton(const std::string& weapon name, unsigned int ammo, Pl
   aver& player) :
        weapon name(weapon name), player(player) {
        this->setLabel(ammo);
        std::string path = WEAPONS_PATH;
        path += weapon name + ".png";
        this->image.set(path);
        this->button.set_image(this->image);
        this->button.set_always_show_image(true);
        this->button.signal_clicked().connect(sigc::mem_fun(*this, &WeaponButton::on
   ClickedButton));
14
   WeaponButton::~WeaponButton() {}
16
   void WeaponButton::onClickedButton() {
19
        this->player.changeWeapon(weapon name);
20
21
   Gtk::Widget& WeaponButton::getButton() {
22
        return this->button;
23
24
   void WeaponButton::setLabel(unsigned int ammo) {
26
        std::string label = "Ammo:\n ";
27
        if (!ammo) {
28
            label += "0";
29
            button.set_sensitive(false);
31
        else if (ammo > 100) {
32
            label += ^{\circ}aM-^{\wedge}HM-^{\wedge};
33
34
35
            label += std::to_string(ammo);
36
        this->button.set_label(label);
37
38
39
```

```
WeaponButton.h
Jun 05. 18 15:28
                                                                             Page 1/1
   #ifndef __CLIENTWEAPONBUTTON_H_
   #define ___CLIENTWEAPONBUTTON_H__
   #include <gtkmm/togglebutton.h>
   #include <gtkmm/image.h>
   #include <string>
   class Player;
   /* Clase que se encarga de mostrar el boton de un arma
    * junto con la informacion correspondiente a esa arma */
   class WeaponButton {
13
       private:
           std::string weapon_name;
14
15
           Player& player;
16
           Gtk::Button button:
17
           Gtk::Image image;
18
       public:
19
20
            /* Constructor */
           WeaponButton(const std::string& weapon name, unsigned int ammo, Player&
21
   player);
22
            /* Destructor */
23
24
            ~WeaponButton();
25
            /* Devuelve el wiget del boton */
26
           Gtk::Widget& getButton();
27
28
            /* Setea el label del boton */
29
            void setLabel(unsigned int ammo);
30
            /* Handler del boton al ser clickeado */
32
            void onClickedButton();
33
34
   };
35
36
   #endif
37
```

```
WeaponView.cpp
Jun 03. 18 12:56
                                                                             Page 1/1
   #include "WeaponView.h"
   #include <glibmm/main.h>
   #include "Player.h"
   #include "WeaponList.h"
   #include "WeaponButton.h"
   WeaponView::WeaponView(WeaponList& weapons, Player& player):
                        weapons(weapons), player(player) {}
   WeaponView::~WeaponView() {}
   void WeaponView::update()
        WeaponList::iterator iter;
        int row = 1, column = 1;
14
15
        for (iter = this->weapons.begin(); iter != this->weapons.end(); iter++) {
           std::unique_ptr<WeaponButton> p(new WeaponButton(iter->second->getName())
   , iter->second->getAmmo(), this->player));
           this->buttons.insert(std::pair<std::string, std::unique_ptr<WeaponButton
   >>(iter->second->getName(), std::move(p)));
           this->window.attach(this->buttons.at(iter->second->getName())->getButton
    (), column, row, 1, 1);
            row++;
20
21
22
   Gtk::Grid& WeaponView::getWindow() {
23
        return this->window;
24
25
26
   void WeaponView::updateAmmo(const Weapon& weapon) {
27
        this->buttons[weapon.getName()]->setLabel(weapon.getAmmo());
28
29
```

```
WeaponView.h
Jun 03. 18 12:56
                                                                            Page 1/1
   #ifndef ___CLIENTWEAPONVIEW_H__
2 #define __CLIENTWEAPONVIEW_H__
   #include <qtkmm/grid.h>
   #include <unordered map>
   #include <memory>
   #include <string>
  class Player;
10 class WeaponList:
11 class WeaponButton;
12 class Weapon;
   /* Clase que se encarga de mostrar los datos de las armas del juego
    * v de almacenar todos los botones de las armas */
15
16
   class WeaponView {
17
       private:
           WeaponList& weapons;
18
           Gtk::Grid window;
19
20
           Player& player;
21
           std::unordered map<std::string, std::unique ptr<WeaponButton>> buttons;
22
       public:
23
            /* Constructor */
24
25
           WeaponView (WeaponList& weapons, Player& player);
26
            /* Destructor */
27
            ~WeaponView();
28
29
30
            /* Actualiza la informacion de todos los botones */
31
32
            void update();
33
            /* Actualiza la informacion de la municion del arma especifica */
34
            void updateAmmo(const Weapon& weapon);
35
36
            /* Devuelve el contenedor de la vista */
37
           Gtk::Grid& getWindow();
38
   };
39
40
   #endif
```

```
WindView.cpp
Jun 02. 18 13:59
                                                                               Page 1/1
    #include "WindView.h"
   #include "Path.h'
   WindView::WindView(): container(false, 7) {
        this->container.pack_start(this->velocity, Gtk::PACK_SHRINK);
        this->container.pack start(this->direction, Gtk::PACK SHRINK);
        this->velocity.set use markup(true);
8
   WindView::~WindView(){}
   void WindView::update(float wind) {
        wind *= 10;
        std::string message = "<span><b><u>Viento</u></b>\n\n";
15
        std::string direction = "right";
16
        if (wind == 0) {
            direction = "no";
17
        } else if (wind < 0) {
18
            wind *=-1;
19
            direction = "left";
20
21
        std::string velocity = std::to string(wind);
        message += velocity.substr(0,4) + "</span>";
23
24
        this->velocity.set_markup(message);
25
        this->direction.set(IMAGES PATH + "arrow " + direction + ".png");
26
27
   Gtk::VBox& WindView::getWindow(){
28
        return this->container:
29
30
```

```
WindView.h
May 30, 18 22:01
                                                                               Page 1/1
    #ifndef ___WINDVIEW_H__
   #define __WINDVIEW_H__
    #include <atkmm/hvbox.h>
    #include <gtkmm/label.h>
    #include <qtkmm/image.h>
8
   class WindView{
        private:
a
10
            Gtk:: VBox container;
11
            Gtk::Label velocity;
12
            Gtk::Image direction;
13
14
        public:
15
            WindView():
16
            ~WindView();
17
            //Actualiza la vista del viento
18
19
            void update(float wind);
20
21
            Gtk::VBox& getWindow();
22
   };
23
    #endif
24
25
```

```
WorldView.cpp
Jun 06. 18 20:39
                                                                             Page 1/2
   #include "WorldView.h"
   #include <gtkmm/adjustment.h>
   #include <qlibmm/main.h>
   #include <giomm/memoryinputstream.h>
   #include "ViewPositionTransformer.h"
   #include "Player.h"
   #include "Math.h"
   #include "Path.h"
   #include "ObjectSizes.h"
   WorldView::WorldView()
        this->container.add_overlay(this->background);
        this->world.set_size(map_width, map_height);
13
14
        this->window.add_events(Gdk::BUTTON_PRESS_MASK);
15
        this->window.add(this->world);
16
        this->container.add_overlay(this->window);
17
        this->water.show(this->world);
18
19
        this->window.get_hadjustment()->set_value(map_width / 2);
20
        this->window.get_vadjustment()->set_value(map_height);
21
   WorldView::~WorldView() {}
24
   void WorldView::moveElement (Gtk::Widget& element, const Position& position, floa
   t width, float height, bool focus) {
       Position newPosition = ViewPositionTransformer(this->world).transformToScree
   nAndMove(position, width, height);
        this->world.move(element, newPosition.getX(), newPosition.getY());
27
        if (focus) {
28
            this->setFocus(element):
29
31
32
   void WorldView::moveScope(Gtk::Widget& scope, Gtk::Widget& worm, int angle) {
33
        float pos_x = this->world.child_property_x(worm).get_value();
        float pos_y = this->world.child_property_y(worm).get_value();
        pos_x += 50 * Math::cosDegrees(angle);
36
       pos_y -= 50 * Math::sinDegrees(angle);
37
       pos_x -= worm.get_width() / 2; // Para que quede referenciado a la mitad de
38
        this->world.move(scope, pos x, pos y);
40
41
   void WorldView::removeElement(Gtk::Widget& element){
        this->world.remove(element);
43
   void WorldView::addElement(Gtk::Widget& element, const Position& position, float
     width, float height, bool focus) {
        Position newPosition = ViewPositionTransformer(this->world).transformToScree
   nAndMove (position, width, height);
        this->world.put(element, newPosition.getX(), newPosition.getY());
        element.show_all();
49
       if (focus) {
50
51
            this->setFocus(element);
52
53
54
   Gtk::ScrolledWindow& WorldView::getWindow() {
55
        return this->window;
56
57
   Gtk::Layout& WorldView::getLayout(){
        return this->world;
60
61
```

```
WorldView.cpp
Jun 06. 18 20:39
                                                                             Page 2/2
   void WorldView::setFocus(Gtk::Widget& element){
       this->window.get hadjustment()->set value(element.get allocation().get x()
    this->window.get_hadjustment()->get_page_size() / 2);
       this->window.get vadjustment()->set value(element.get allocation().get v()
65
    this->window.get vadjustment()->get page size() / 2);
66
67
   void WorldView::setBackgroundImage(const Buffer& image) {
68
       sigc::slot<br/>bool> my slot = sigc::bind(sigc::mem fun(*this, &WorldView::setBa
   ckgroundImageCallBack), image);
70
       Glib::signal_idle().connect(my_slot);
71 }
72
73
   bool WorldView::setBackgroundImageCallBack(Buffer image) {
74
       auto screen = this->container.get screen();
75
       size_t screen_width = screen->get_width();
       size_t screen_height = screen->get_height();
76
       auto pixbuf = Gio::MemoryInputStream::create();
77
78
       pixbuf->add_data(image.getPointer(), image.getMaxSize());
79
       auto aux = Gdk::Pixbuf::create from stream (pixbuf);
       size t img width = aux->get width();
       size t img height = aux->get height();
81
       for (size_t x = 0; x < screen_width; x += img_width) {</pre>
82
83
            for (size_t y = 0; y < screen_height; y += img_height) {</pre>
                Gtk:: Image background image (aux);
84
                background_image.show();
85
                this->background.put(background image, x, y);
86
                this->background_images.push_back(std::move(background_image));
87
88
89
       return false:
90
91
92
   Gtk::Container& WorldView::getContainer() {
93
94
       return this->container;
95
```

```
WorldView.h
Jun 06. 18 20:17
                                                                              Page 1/2
    #ifndef ___WORLDVIEW_H__
   #define WORLDVIEW H
   #include <gtkmm/widget.h>
   #include <qtkmm/layout.h>
   #include <gtkmm/hvbox.h>
   #include <gtkmm/scrolledwindow.h>
   #include <qtkmm/overlay.h>
   #include <string>
   #include "Position.h"
   #include "Water.h"
   #include "Buffer.h"
   class Player;
   /* Clase que se encarga de mostrar objetos en posiciones
    * especificas, moverlos y eliminarlos de la vista*/
   class WorldView{
       private:
19
20
            Gtk::Overlay container;
21
            Gtk::Layout background;
22
            Gtk::Layout world;
            Gtk::ScrolledWindow window;
23
24
            std::vector<Gtk::Image> background images;
25
            Water water:
26
            bool setBackgroundImageCallBack(Buffer image);
27
28
        public:
29
            /* Constructor */
30
            WorldView():
31
32
            /* Destructor */
33
            ~WorldView();
34
35
36
            /* Setea la imagen de fondo */
            void setBackgroundImage(const Buffer& image);
37
38
            /* Mueve el elemento pasado a la posicion especificada */
39
            void moveElement (Gtk::Widget& element, const Position& position, float w
   idth, float height, bool focus = false);
            /* Mueve la mira a la posicion correspondiente para que tenga el angulo
42
             * especificado por parametro */
43
44
            void moveScope (Gtk::Widget& scope, Gtk::Widget& worm, int angle);
45
            /* Remueve el elemento de la vista */
46
            void removeElement(Gtk::Widget& element);
47
            /* Agrega un elemento a la vista en la posicion especificada */
            void addElement (Gtk::Widget& element, const Position& position, float wi
   dth, float height, bool focus = false);
            /* Devuelve la vista del scrolledWindow */
52
            Gtk::ScrolledWindow& getWindow();
53
54
            /* Devuelve el container */
55
            Gtk::Container& getContainer();
56
57
            /* Devuelve la vista del Layout */
58
            Gtk::Layout& getLayout();
59
61
            /* Realiza focus en el elemento pasado */
            void setFocus(Gtk::Widget& element);
62
63
   };
```

Jun 06, 18 20:17 **WorldView.h** Page 2/2

```
Turn.cpp
May 31, 18 12:08
                                                                             Page 1/1
   #include "Turn.h"
   #include <glibmm/main.h>
   #include "Player.h"
   const int TIMER = 60;
   const int REDUCTION TIME = 3;
   const int LIMIT TIME = 10;
   Turn::Turn(Player& player, TurnLabel& time_label):
        actual_time(TIMER), player(player), time_label(time_label){}
   Turn::~Turn() {}
14 bool Turn::startCallBack() {
15
        this->time_label.setTime(this->actual_time);
16
        if (this->actual_time <= LIMIT_TIME) {</pre>
17
            this->player.playTickTime();
18
19
        if (this->actual_time == 0)
20
            this->player.endTurn();
21
22
        this->actual time--;
        return this->actual_time >= 0;
23
24
25
26
   void Turn::start() {
        this->actual_time = TIMER;
        this->my_connection = Glib::signal_timeout().connect(sigc::mem_fun(*this, &T
   urn::startCallBack), 1000);
29
30
   void Turn::reduceTime() {
       this->actual_time = REDUCTION_TIME;
32
33
34
   void Turn::stop() {
35
       if (this->my_connection.connected()) {
            this->my_connection.disconnect();
37
            this->player.endTurn();
38
39
40
```

Turn.h May 31, 18 12:08 Page 1/1 #ifndef ___CLIENTTURN_H__ 2 #define __CLIENTTURN_H__ #include "TurnLabel.h" class Player; /* Clase que se encarga de contar el tiempo del turno */ class Turn { 10 private: 11 int actual time; 12 Player& player; 13 TurnLabel& time_label; sigc::connection my_connection; 14 15 16 /* Callback de start */ 17 bool startCallBack(); 18 public: 19 20 /* Constructor */ 21 Turn(Player& player, TurnLabel& time label); 22 /* Destructor */ 23 ~Turn(); 24 25 26 /* Comienza la cuenta regresiva del turno actualizando el 27 * label que muestra el tiempo */ 28 void start(); 29 30 /* Reduce el tiempo restante del turno a 3 segundos */ 31 void reduceTime(); 32 33 /* Detiene el contador y finaliza el turno */ 34 void stop(); 35 36 }; 38 #endif

```
BulletView.cpp
May 31, 18 12:08
                                                                             Page 1/1
   #include "BulletView.h"
   #include "ObjectSizes.h"
   BulletView::BulletView(WorldView& worldView, std::string weapon, Position pos):
       Viewable (worldView), weapon name (std::move (weapon)) {
        std::string path(BULLETS PATH);
       path += this->weapon name;
       path += ".png";
10
        this->image.set(path);
        this->addToWorld(pos, weapon size, weapon size);
14
   BulletView::~BulletView() {}
16
   BulletView::BulletView(BulletView&& other): Viewable(std::move(other)),
        image(std::move(other.image)), weapon_name(std::move(other.weapon_name)) {}
   void BulletView::updateData(const Position& new_pos) {
19
20
        this->move(new_pos, weapon_size, weapon_size);
21
   Gtk::Widget& BulletView::getWidget() {
       return this->image;
24
25
26
   std::string BulletView::getName() {
        return this->weapon name;
28
29
30
```

```
BulletView.h
May 31, 18 12:08
                                                                             Page 1/1
   #ifndef __CLIENTBULLETVIEW_H__
2 #define __CLIENTBULLETVIEW_H__
   #include <gtkmm/widget.h>
   #include <gtkmm/image.h>
   #include <string>
   #include "Viewable.h"
   /* Clase que se encarga de controlar la vista de las balas */
   class BulletView: public Viewable{
       private:
12
           Gtk::Image image;
13
           std::string weapon_name;
14
15
       public:
16
            /* Constructor */
17
           BulletView(WorldView& worldView, std::string weapon, Position pos);
18
            /* Destructor */
19
20
            ~BulletView();
21
22
            /* Constructor por movimient */
           BulletView(BulletView&& other);
23
24
25
            /* Actualiza la posicion de la bala en la vista */
            void updateData(const Position& new pos);
26
27
            /* Devuelve el contenedor de la bala */
28
           Gtk::Widget& getWidget() override;
29
30
            /* Devuelve el nombre del arma de la bala */
31
32
            std::string getName();
33
   };
34
35
36 #endif
```

```
GirderView.cpp
Jun 05. 18 14:07
                                                                            Page 1/1
   #include "GirderView.h"
   #include "GirderSize.h"
   GirderView::GirderView(WorldView& worldView, size t size, Position pos, int rota
       Viewable(worldView), size(size), rotation(rotation){
        std::string path(GIRDER PATH);
       path += std::to_string(size);
       path += " ";
       path += std::to string(rotation);
       path += ".png";
12
        this->image.set(path);
13
        float width = GirderSize::getGirderWidthMeters(size, rotation);
14
        float height = GirderSize::getGirderHeightMeters(size, rotation);
15
        this->addToWorld(pos, width, height);
16
   GirderView::~GirderView(){}
20
   GirderView::GirderView(GirderView&& other): Viewable(std::move(other)),
        image(std::move(other.image)), size(other.size), rotation(other.rotation){}
22
   Gtk::Widget& GirderView::getWidget(){
23
24
        return this->image;
25
26
```

GirderView.h Jun 05. 18 14:07 Page 1/1 #ifndef ___GIRDERVIEW_H__ 2 #define __GIRDERVIEW_H_ #include <gtkmm/widget.h> #include <gtkmm/image.h> #include <string> #include "Viewable.h" /* Clase que se encaga de controlar la vista de las vigas */ class GirderView: public Viewable{ private: 12 Gtk::Image image; 13 int size; int rotation; 14 15 16 public: /* Constructor */ 17 GirderView(WorldView& worldView, size_t size, Position pos, int rotation 18); 19 20 /* Destructor */ 21 ~GirderView(); 22 /* Constructor por movimiento */ 23 24 GirderView(GirderView&& other); 25 /* Devuelve el contenedor de la viga */ 26 Gtk::Widget& getWidget() override; 27 28 29 30 #endif

```
Viewable.cpp
May 28, 18 18:21
                                                                             Page 1/1
   #include "Viewable.h"
   Viewable:: Viewable (WorldView& worldView): worldView (worldView), has focus (false)
   Viewable::~Viewable(){}
   void Viewable::move(const Position& pos, float width, float height) {
       this->worldView.moveElement(this->getWidget(), pos, width, height, this->has
    focus);
   void Viewable::removeFromWorld() {
       this->worldView.removeElement(this->getWidget());
12
13
   void Viewable::addToWorld(const Position& pos, float width, float height) {
       this->worldView.addElement(this->getWidget(), pos, width, height, this->has_
17
   Viewable::Viewable(Viewable&& other): worldView(other.worldView), has focus(other)
   r.has focus) {}
20
21
   void Viewable::setFocus(bool focus) {
        this->has focus = focus;
22
23
24
   bool Viewable::hasFocus() const {
       return this->has focus;
27
```

```
Viewable.h
May 31, 18 12:08
                                                                             Page 1/1
   #ifndef __VIEWABLE_H__
2 #define ___VIEWABLE_H__
   #include <qtkmm/widget.h>
   #include "WorldView.h"
   #include "Position.h"
   #include "Path.h"
   /* Clase que se encarga de controlar los objetos visuales */
   class Viewable{
       private:
12
            WorldView& worldView;
13
            bool has_focus;
14
15
       protected:
16
            /* Agrega al objeto visual a la vista */
17
            void addToWorld(const Position& pos, float width, float height);
18
            /* Mueve al objeto visual a la posicion especificada */
19
20
            void move (const Position& pos, float width, float height);
21
22
        public:
            /* Constructor */
23
            Viewable (WorldView& worldView);
24
25
            /* Destructor */
26
            virtual ~Viewable();
27
28
            /* Constructor por movimiento */
29
            Viewable (Viewable & other):
30
31
            /* Devuelve el contenedor del objeto visual */
32
            virtual Gtk::Widget& getWidget() = 0;
33
34
            /* Remueve al objeto visual de la vista */
35
            void removeFromWorld();
36
37
            /* Establece si al objeto visual se le puede hacer focus o no */
38
            void setFocus(bool focus);
39
40
            /* Devuelve true si el objeto visual es focuseable */
41
            bool hasFocus() const;
43
   #endif
```

```
WormLifeView.cpp
Jun 06. 18 20:08
                                                                             Page 1/1
   #include "WormLifeView.h"
   const std::string begining("<span color='");</pre>
   const std::string middle("'><b>");
   const std::string ending("</b></span>");
   WormLifeView::WormLifeView(int life, const std::string& color): color(color) {
        this->label.set use markup(true);
        this->updateLife(life);
10
   WormLifeView::~WormLifeView(){}
   WormLifeView::WormLifeView(WormLifeView&& other):
       label(std::move(other.label)), color(std::move(other.color)){}
   void WormLifeView::updateLife(int life) {
       this->label.override_background_color(Gdk::RGBA(this->color));
        this->label.set_markup(begining + "white" + middle + std::to_string(life) + e
   nding);
20
   Gtk::Widget& WormLifeView::getWidget(){
       return this->label;
23
24
```

```
WormLifeView.h
May 27, 18 21:56
   #ifndef __WORMLIFEVIEW_H__
   #define WORMLIFEVIEW H
   #include <qtkmm/label.h>
    /* Clase que se encarga de controlar el label de la vida
    * del worm */
   class WormLifeView{
       private:
10
            Gtk::Label label;
11
            std::string color;
12
13
        public:
14
            /* Constructor */
            WormLifeView(int life, const std::string& color);
15
16
17
            /* Destructor */
            ~WormLifeView();
18
19
20
            /* Constructor por movimiento */
21
            WormLifeView(WormLifeView&& other);
22
            /* Actualiza el label de vida del worm */
23
            void updateLife(int life);
24
25
            /* Devuelve el contenedor de la vida */
26
            Gtk::Widget& getWidget();
27
   };
28
29
30
   #endif
```

```
WormView.cpp
Jun 06. 18 20:08
                                                                              Page 1/2
   #include "WormView.h"
   #include <string>
   #include <qlibmm/main.h>
   #include "ObjectSizes.h"
   #include "WeaponNames.h"
#include "GamePlayers.h"
8 WormView::WormView(WorldView& worldView, int life, char dir, Position pos, int p
        Viewable (worldView), player id(player id), life(life), is moving(false),
        last position(Position(-1, -1)), label(life, colors[player id]),
        walkingAnimation(&this->image), weaponAnimation(DEFAULT_WEAPON, &this->image
   ) {
12
            this->worm.attach(this->label.getWidget(), 0, 0, 1, 1);
13
            this->worm.attach(this->image, 0, 1, 1, 1);
14
            this->walkingAnimation.setStaticImage();
15
            this->addToWorld(pos, worm_size, worm_size + 0.5);
16
17
18
   WormView::~WormView(){}
   WormView::WormView(WormView&& other): Viewable(std::move(other)), player id(othe
   r.player id),
        life (other.life), is_moving (other.is_moving),
21
22
        last position (other.last position), label(std::move(other.label)),
        image(std::move(other.image)),
23
        worm(std::move(other.worm)), walkingAnimation(std::move(other.walkingAnimati
24
        weaponAnimation(std::move(other.weaponAnimation)) {
25
        this->weaponAnimation.updateWormImage(&this->image);
26
        this->walkingAnimation.updateWormImage(&this->image);
27
28
   void WormView::updateData(int new_life, char new_dir, const Position& new_pos, b
   ool colliding, bool is_current_worm, bool has_shot)
       if (new_life != this->life) {
31
            this->label.updateLife(new_life);
32
33
        this->life = new life;
34
        this->is moving = !(this->last_position == new_pos);
35
        this->last position = new pos;
36
        this->setNewImage(new dir, colliding, is current worm, has shot);
        this->move(new pos, worm size, worm size + 0.5);
38
39
40
   void WormView::updateScope(int angle) {
41
        this->weaponAnimation.changeAngle(angle, this->getDir());
42
43
44
45
   void WormView::changeWeapon(const std::string& weapon) {
        this->weaponAnimation.changeWeapon(weapon, this->getDir());
47
   void WormView::setNewImage(char dir, bool colliding, bool is_current_worm, bool
   has shot) {
        if (is current worm) {
            if (!this->is moving && !has shot && colliding) {
51
                this->weaponAnimation.setWeaponImage(dir);
52
              else if (colliding) {
53
                this->walkingAnimation.setMovementImage(dir);
54
55
            return;
57
        this->walkingAnimation.setStaticImage();
58
59
```

Page 1/1

```
WormView.cpp
                                                                              Page 2/2
Jun 06. 18 20:08
   Gtk::Widget& WormView::getWidget() {
       return this->worm;
62
63
   Gtk::Tmage& WormView::getTmage() {
65
       return this->image;
66
67
68
   int WormView::getLife() const{
60
70
       return this->life;
71
72
73
   char WormView::getDir() const
74
       return this->walkingAnimation.getDir();
75
76
77
   int WormView::getPlayerId() const{
       return this->player_id;
78
79
80
81
   bool WormView::isMoving() const{
       return this->is moving;
82
83
85
   void WormView::setVictorv() {
       this->image.set(VICTORY ANIMATION);
86
87
88
   void WormView::weaponShoot(const std::string& weapon) {
89
       this->weaponAnimation.weaponShootAnimation(weapon, this->getDir());
90
91
   void WormView::resetFocus() {
93
       this->is_moving = false;
       this->setFocus(false);
95
96
       this->walkingAnimation.setStaticImage();
97 }
```

```
WormView.h
Jun 06. 18 20:08
                                                                             Page 1/2
    #ifndef ___WORMVIEW_H__
   #define __WORMVIEW_H
   #include <gtkmm/widget.h>
   #include <gtkmm/image.h>
   #include <gtkmm/grid.h>
   #include <qdkmm/pixbuf.h>
   #include <vector>
   #include "Viewable.h"
   #include "WormLifeView.h"
   #include "WalkingAnimation.h"
   #include "WeaponAnimation.h"
   #define DIR_RIGHT 1
   #define DIR LEFT -1
   /* Clase que se encarga de controlar la vista de los worms */
   class WormView: public Viewable {
       private:
20
            int player_id;
21
            int life;
22
            bool is moving;
            Position last_position;
23
24
            WormLifeView label;
25
            Gtk:: Tmage image:
            Gtk::Grid worm;
26
            WalkingAnimation walkingAnimation;
27
            WeaponAnimation weaponAnimation;
28
29
            /* Actualiza la imagen del worm a la correspondiente segun las
30
             * condiciones en las que se encuentra este */
31
            void setNewImage(char dir, bool colliding, bool is_current_worm, bool ha
   s_shot);
33
            /* Cambia la imagen actual por la del arma actual */
34
            void setWeaponImage();
35
36
            /* Actualiza las imagenes de las armas */
37
            void updateWeaponImage();
38
            bool batHitCallBack(std::vector<Glib::RefPtr<Gdk::Pixbuf>>::iterator& it
   er, const int width);
        public:
43
            /* Constructor */
44
            WormView(WorldView& worldView, int life, char dir, Position pos, int pla
   yer_id);
            /* Destructor */
47
            ~WormView();
48
49
            /* Constructor por movimiento */
50
51
            WormView(WormView&& other);
52
53
            /* Actualiza la posicion v vida del worm */
            void updateData(int new_life, char new_dir, const Position& new_pos, boo
   l colliding, bool is_current_worm, bool has_shot);
            /* Actualiza la imagen del arma con el angulo actual */
            void updateScope(int angle);
59
            /* Actualiza el arma del worm y cambia la imagen */
60
61
            void changeWeapon(const std::string &weapon);
```

```
WormView.h
Jun 06. 18 20:08
                                                                            Page 2/2
            /* Devuelve la direccion del worm */
           char getDir() const;
64
65
            /* Elimina la imagen del arma del worm */
66
            void removeWeaponImage();
67
68
            /* Devuelve la vida del worm */
69
            int getLife() const;
70
71
            /* Devuelve el id del player que controla al worm */
72
            int getPlayerId() const;
73
74
            /* Devuelve el contenedor donde se encuentra la vista del worm */
75
           Gtk::Widget& getWidget() override;
76
77
78
            /* Devuelve la imagen que contiene al worm */
79
           Gtk::Image& getImage();
80
            /* Cambia la imagen del worm por la animacion del worm
81
82
            * festejando la victoria */
83
            void setVictory();
            /* Devuelve true si el gusano se esta moviendo */
85
           bool isMoving() const;
86
87
88
            /* Realiza la animacion del disparo del arma */
            void weaponShoot(const std::string& weapon);
89
90
            /* Resetea el focus del gusano */
91
            void resetFocus();
92
   };
93
   #endif
```

```
DistanceWeapon.cpp
May 26, 18 12:13
                                                                            Page 1/1
   #include "DistanceWeapon.h"
   DistanceWeapon::DistanceWeapon(std::string name, int ammo, bool time) :
       Weapon (name, ammo) {
        this->has_Scope = true;
       this->is Timed = time;
   DistanceWeapon::~DistanceWeapon() {}
11 DistanceWeapon::DistanceWeapon(DistanceWeapon&& other) : Weapon(std::move(other)
12
   bool DistanceWeapon::hasVariablePower() const{
13
14
       return true:
15
16
```

DistanceWeapon.h May 27, 18 21:56 Page 1/1 #ifndef __CLIENTDISTANCEWEAPON_H__ #define __CLIENTDISTANCEWEAPON_H__ #include "Weapon.h" /* Clase que se encarga de representar a las armas de distancia */ class DistanceWeapon: public Weapon{ public: /* Constructor */ 9 10 DistanceWeapon(std::string name, int ammo, bool time = false); 12 /* Destructor */ 13 ~DistanceWeapon(); 14 15 /* Constructor por movimiento */ DistanceWeapon(DistanceWeapon&& other); 16 17 18 19 /* Devuelve true si el arma tiene potencia variable */ 20 bool hasVariablePower() const override; 21 23 #endif

MeleeWeapon.h May 27, 18 21:56 Page 1/1 #ifndef __CLIENTMELEEWEAPON_H_ #define __CLIENTMELEEWEAPON_H_ #include "Weapon.h" /* Clase que se encarga de representar las armas de cuerpo a cuerpo */ class MeleeWeapon : public Weapon { public: /* Constructor */ 10 MeleeWeapon(std::string name, int ammo, bool scope, bool time = false); 12 /* Destructor */ ~MeleeWeapon() {} 13 14 15 /* Constructor por movimiento */ 16 MeleeWeapon (MeleeWeapon&& other); 17 18 19 #endif

```
May 26, 18 12:13

#include "AirAttack.h"
#include "WeaponNames.h"

AirAttack::AirAttack(int ammo): SelfDirectedWeapon(AIR_ATTACK_NAME, ammo) {}

AirAttack::~AirAttack() {}

AirAttack::AirAttack(AirAttack& other): SelfDirectedWeapon(std::move(other)) {}

AirAttack::AirAttack(AirAttack& other): SelfDirectedWeapon(std::move(other)) {}
```

```
May 27, 18 21:56
                                       AirAttack.h
                                                                            Page 1/1
   #ifndef __CLIENTAIRATTACK_H__
   #define __CLIENTAIRATTACK_H_
   #include "SelfDirectedWeapon.h"
   /* Clase que representa al arma AirStrike */
6
    class AirAttack: public SelfDirectedWeapon {
        public:
            /* Constructor */
 9
10
            AirAttack(int ammo);
11
12
            /* Destructor */
13
            ~AirAttack();
14
15
            /* Constructor por movimiento */
16
            AirAttack (AirAttack&& other);
17
   };
18
19 #endif
```

```
[75.42] Taller de programacion
May 26, 18 12:13
                                      Banana.cpp
                                                                            Page 1/1
    #include "Banana.h"
   #include "WeaponNames.h"
   Banana::Banana(int ammo) : DistanceWeapon(BANANA_NAME, ammo, true) {}
   Banana::~Banana() {}
   Banana::Banana(Banana&& other) : DistanceWeapon(std::move(other)) { }
```

```
May 27, 18 21:56
                                        Banana.h
                                                                            Page 1/1
   #ifndef __CLIENTBANANA_H__
2 #define __CLIENTBANANA_H_
   #include "DistanceWeapon.h"
   /* Clase que representa al arma Banana */
6
    class Banana: public DistanceWeapon {
        public:
            /* Constructor */
 9
10
            Banana (int ammo);
11
            /* Destructor */
12
13
            ~Banana();
14
            /* Constructor por movimiento */
15
16
            Banana (Banana&& other);
17
   };
18
19 #endif
```

```
[75.42] Taller de programacion
May 26, 18 12:13
                                         Bat.cpp
                                                                            Page 1/1
    #include "Bat.h"
   #include "WeaponNames.h"
   Bat::Bat(int ammo): MeleeWeapon(BAT_NAME, ammo, true) {}
6 Bat::~Bat() {}
   Bat::Bat(Bat&& other) : MeleeWeapon(std::move(other)) {}
```

```
May 31, 18 12:08
                                          Bat.h
                                                                            Page 1/1
   #ifndef __CLIENTBAT_H__
2 #define __CLIENTBAT_H_
   #include "MeleeWeapon.h"
   /* Clase que representa al arma Bat de baseball */
6
    class Bat: public MeleeWeapon {
        public:
            /* Constructor */
 9
10
           Bat(int ammo);
11
            /* Destructor */
12
13
            ~Bat();
14
15
            /* Constructor por movimiento */
16
            Bat (Bat&& other);
17
   };
18
19 #endif
```

```
May 26, 18 12:13

Bazooka.cpp

Page 1/1

#include "Bazooka.h"

#include "WeaponNames.h"

Bazooka::Bazooka(int ammo) : DistanceWeapon(BAZOOKA_NAME, ammo) {}

Bazooka::~Bazooka(){}

Bazooka::Bazooka(Bazooka&& other) : DistanceWeapon(std::move(other)) {}

Bazooka::Bazooka(Bazooka&& other) : DistanceWeapon(std::move(other)) {}
```

```
May 27, 18 21:56
                                       Bazooka.h
                                                                            Page 1/1
   #ifndef __CLIENTBAZOOKA_H_
   #define __CLIENTBAZOOKA_H_
   #include "DistanceWeapon.h"
   /* Clase que representa al arma Bazooka */
6
    class Bazooka: public DistanceWeapon {
        public:
            /* Constructor */
10
            Bazooka (int ammo);
12
            /* Destructor */
13
            ~Bazooka();
14
            /* Constructor por movimiento */
15
16
            Bazooka (Bazooka&& other);
17
18
19 #endif
```

```
[75.42] Taller de programacion
May 26, 18 12:13
                                     Dynamite.cpp
                                                                            Page 1/1
    #include "Dynamite.h"
   #include "WeaponNames.h"
   Dynamite::Dynamite(int ammo): MeleeWeapon(DYNAMITE_NAME, ammo, false, true) {}
   Dynamite::~Dynamite() {}
   Dynamite::Dynamite(Dynamite&& other) : MeleeWeapon(std::move(other)) {}
```

```
Dynamite.h
May 27, 18 21:56
                                                                            Page 1/1
   #ifndef __CLIENTDYNAMITE_H__
   #define __CLIENTDYNAMITE_H_
    #include "MeleeWeapon.h"
   /* Clase que representa al arma Dinamita */
    class Dynamite: public MeleeWeapon {
        public:
            /* Constructor */
10
            Dynamite(int ammo);
12
            /* Destructor */
13
            ~Dynamite();
14
            /* Constructor por movimiento */
15
16
            Dynamite (Dynamite&& other);
17
18
19 #endif
```

```
May 26, 18 12:13 GreenGrenade.cpp Page 1/1

#include "GreenGrenade.h"
#include "WeaponNames.h"

GreenGrenade::GreenGrenade(int ammo):
DistanceWeapon(GREEN_GRENADE_NAME, ammo, true) {}

GreenGrenade::~GreenGrenade() {}

GreenGrenade::GreenGrenade() {}

GreenGrenade::GreenGrenade(GreenGrenade&& other): DistanceWeapon(std::move(other)) {}

However the page 1/1
```

```
GreenGrenade.h
May 31, 18 12:08
                                                                           Page 1/1
   #ifndef __CLIENTGREENGRENADE_H_
   #define __CLIENTGREENGRENADE_H_
    #include "DistanceWeapon.h"
   /* Clase que representa al arma Granada verde */
    class GreenGrenade: public DistanceWeapon {
        public:
            /* Constructor */
10
            GreenGrenade (int ammo);
12
            /* Destructor */
            ~GreenGrenade();
13
14
15
            /* Constructor por movimiento */
16
            GreenGrenade (GreenGrenade&& other);
17
18
19 #endif
```

```
May 26, 18 12:13 HolyGrenade.cpp Page 1/1

#include "HolyGrenade.h"
#include "WeaponNames.h"

HolyGrenade::HolyGrenade(int ammo):
DistanceWeapon(HOLY_GRENADE_NAME, ammo, true) {}

HolyGrenade::HolyGrenade(){}

HolyGrenade::HolyGrenade(HolyGrenade&other): DistanceWeapon(std::move(other))
{}
```

```
HolyGrenade.h
May 31, 18 12:08
                                                                            Page 1/1
   #ifndef ___CLIENTHOLYGRENADE_H__
   #define __CLIENTHOLYGRENADE_H_
    #include "DistanceWeapon.h"
   /* Clase que representa al arma Granada santa */
    class HolyGrenade: public DistanceWeapon {
        public:
            /* Constructor */
10
            HolyGrenade(int ammo);
12
            /* Destructor */
            ~HolyGrenade();
13
14
15
            /* Constructor por movimiento */
16
            HolyGrenade (HolyGrenade&& other);
17
18
19 #endif
```

```
[75.42] Taller de programacion
May 26, 18 12:13
                                       Mortar.cpp
                                                                            Page 1/1
    #include "Mortar.h"
   #include "WeaponNames.h"
   Mortar::Mortar(int ammo): DistanceWeapon(MORTAR_NAME, ammo, false) {}
   Mortar::~Mortar() {}
   Mortar::Mortar(Mortar&& other) : DistanceWeapon(std::move(other)) { }
```

```
May 27, 18 21:56
                                        Mortar.h
                                                                            Page 1/1
   #ifndef __CLIENTMORTAR_H_
   #define __CLIENTMORTAR_H_
    #include "DistanceWeapon.h"
   /* Clase que representa al arma Mortero */
    class Mortar: public DistanceWeapon {
        public:
            /* Constructor */
10
           Mortar(int ammo);
12
            /* Destructor */
13
            ~Mortar();
14
15
            /* Constructor por movimiento */
            Mortar (Mortar&& other);
16
17
   };
18
19 #endif
```

```
May 26, 18 12:13 RedGrenade.cpp Page 1/1

#include "RedGrenade.h"
#include "WeaponNames.h"

RedGrenade::RedGrenade(int ammo):
DistanceWeapon(RED_GRENADE_NAME, ammo, true) {}

RedGrenade::~RedGrenade() {}

RedGrenade::RedGrenade() {}

RedGrenade::RedGrenade() {}
```

```
RedGrenade.h
May 31, 18 12:08
                                                                           Page 1/1
   #ifndef __CLIENTREDGRENADE_H_
   #define __CLIENTREDGRENADE_H_
    #include "DistanceWeapon.h"
   /* Clase que representa al arma Granada roja */
    class RedGrenade: public DistanceWeapon {
        public:
            /* Constructor */
10
            RedGrenade (int ammo);
12
            /* Destructor */
13
            ~RedGrenade();
14
15
            /* Constructor por movimiento */
16
            RedGrenade (RedGrenade&& other);
17
18
19 #endif
```

```
May 26, 18 12:13 Teleportation.cpp Page 1/1

#include "Teleportation.h"
#include "WeaponNames.h"

Teleportation::Teleportation(int ammo): SelfDirectedWeapon(TELEPORT_NAME, ammo)
{}

Teleportation::~Teleportation() {}

Teleportation::Teleportation(Teleportation&& other): SelfDirectedWeapon(std::move(other)) {}

Teleportation:Teleportation(Teleportation&& other): SelfDirectedWeapon(std::move(other)) {}
```

```
Teleportation.h
May 31, 18 12:08
                                                                             Page 1/1
   #ifndef __CLIENTTELEPORTATION_H_
   #define __CLIENTTELEPORTATION_H__
   #include "SelfDirectedWeapon.h"
   /* Clase que representa al arma Teletransportador */
6
    class Teleportation: public SelfDirectedWeapon {
        public:
            /* Constructor */
10
            Teleportation(int ammo);
12
            /* Destructor */
13
            ~Teleportation();
14
15
            /* Constructor por movimiento */
16
            Teleportation (Teleportation & other);
17
18
19 #endif
```

```
May 26, 18 12:13

SelfDirectedWeapon.cpp

Page 1/1

include "SelfDirectedWeapon.h"

SelfDirectedWeapon::SelfDirectedWeapon(std::string name, int ammo) : Weapon(name, ammo) {}

SelfDirectedWeapon::~SelfDirectedWeapon() {}

SelfDirectedWeapon::SelfDirectedWeapon(SelfDirectedWeapon&& other) : Weapon(std::move(other)) {}

bool SelfDirectedWeapon::isSelfDirected() const{
return true;
}

bool SelfDirectedWeapon::isSelfDirected() const{
const()
cons
```

SelfDirectedWeapon.h May 31, 18 12:08 Page 1/1 #ifndef ___SELFDIRECTEDWEAPON_H__ #define __SELFDIRECTEDWEAPON_H_ #include "Weapon.h" /* Clase que representa las armas teledirigidas */ class SelfDirectedWeapon: public Weapon{ public: /* Constructor */ a 10 SelfDirectedWeapon(std::string name, int ammo); 12 /* Destructor */ 13 ~SelfDirectedWeapon(); 14 15 /* Constructor por movimiento */ 16 SelfDirectedWeapon (SelfDirectedWeapon&& other); 17 /* Devuelve true si es teledirigida */ 18 bool isSelfDirected() const override; 19 20 21 #endif

```
Weapon.cpp
May 26, 18 12:13
                                                                              Page 1/1
   #include "Weapon.h"
   Weapon::Weapon(std::string name, int ammo) :
       name (name), ammo (ammo), has_Scope (false), is_Timed (false) {}
   Weapon::~Weapon() {}
   Weapon::Weapon(Weapon&& other) {
        this->name = std::move(other.name);
        this->ammo = std::move(other.ammo);
        this->has Scope = std::move(other.has Scope);
12
        this->is_Timed = std::move(other.is_Timed);
13
15
   Weapon& Weapon::operator=(Weapon&& other) {
16
        this->name = std::move(other.name);
       this->ammo = std::move(other.ammo);
17
        this->has_Scope = std::move(other.has_Scope);
18
19
        this->is_Timed = std::move(other.is_Timed);
20
        return *this:
21
   bool Weapon::hasScope() const{
        return this->has_Scope;
24
25
26
   bool Weapon::isSelfDirected() const{
        return false;
28
29
30
   bool Weapon::isTimed() const{
        return this->is_Timed;
33
34
   bool Weapon::hasVariablePower() const{
35
36
        return false;
37
   const std::string& Weapon::getName() const{
39
        return this->name;
40
41
   void Weapon::shoot() {
       if (this->ammo <= 100)
44
45
            this->ammo--;
46
   bool Weapon::hasAmmo() const{
       return this->ammo > 0;
49
50
   unsigned int Weapon::getAmmo() const{
       return this->ammo;
53
54
55
```

```
Weapon.h
May 27, 18 21:56
                                                                             Page 1/1
   #ifndef __CLIENTWEAPON_H__
   #define ___CLIENTWEAPON_H__
   #include <string>
6
   /* Clase que se encarga de representar a las armas del juego */
    class Weapon {
       protected:
            std::string name;
10
            unsigned int ammo;
            bool has Scope;
12
            bool is_Timed;
13
14
        public:
15
            /* Constructor */
16
            Weapon(std::string name, int ammo);
17
            /* Destructor */
18
19
            ~Weapon();
20
21
            /* Constructor por movimiento */
22
            Weapon (Weapon&& other);
23
            /* Operador = por movimiento */
24
25
            Weapon& operator=(Weapon&& other);
26
27
            /* Devuelve true si el arma tiene mira */
28
            virtual bool hasScope() const;
29
30
            /* Devuelve true si el arma es teledirigida */
31
            virtual bool isSelfDirected() const;
32
33
            /* Devuelve true si el arma es por tiempo */
34
            virtual bool isTimed() const;
35
36
            /* Devuelve true si el arma tiene potencia variable */
37
            virtual bool hasVariablePower() const;
38
39
            /* Devuelve el nombre del arma */
40
            virtual const std::string& getName() const;
41
42
            /* Disminuve la cantidad de municiones del arma */
43
            virtual void shoot();
44
45
            /* Devuelve true si el arma tiene balas */
46
            virtual bool hasAmmo() const;
47
48
            /* Devuelve la cantidad de balas */
49
            unsigned int getAmmo() const;
50
   #endif
```

```
WeaponList.cpp
May 27, 18 21:56
                                                                             Page 1/1
   #include "WeaponList.h"
   #include "WeaponNames.h"
   WeaponList::WeaponList(): current_weapon(DEFAULT_WEAPON) {}
   WeaponList::~WeaponList() {}
   void WeaponList::add(std::string weapon, int ammo) {
        WeaponsFactory factory;
        this->weapons.insert(std::pair<std::string, weapon ptr>(weapon, std::move(fa
   ctory.createWeapon(weapon, ammo))));
12
13
   void WeaponList::changeWeapon(std::string weapon) {
14
        this->current_weapon = weapon;
15
16
   Weapon& WeaponList::getCurrentWeapon() {
17
        return *this->weapons.at(this->current_weapon);
18
19
20
   WeaponList::iterator WeaponList::begin() {
        return this->weapons.begin();
22
23
24
   WeaponList::iterator WeaponList::end() {
25
        return this->weapons.end();
27
28
```

```
WeaponList.h
May 31, 18 12:08
                                                                              Page 1/1
    #ifndef ___CLIENTWEAPONLIST_H__
   #define ___CLIENTWEAPONLIST_H__
    #include <map>
    #include "Weapon.h"
    #include "WeaponsFactory.h"
    /* Clase que se encarga de almacenar las armas del juego */
    class WeaponList {
        private:
10
11
            typedef std::map<std::string, weapon ptr> WeaponsList;
12
            WeaponsList weapons;
13
            std::string current_weapon;
14
15
        public:
16
            /* Constructor */
17
            WeaponList();
18
            /* Destructor */
19
20
            ~WeaponList();
21
22
            /* Agrega un arma a la lista */
23
            void add(std::string weapon, int ammo);
24
25
            /* Devuelve el arma actual */
26
            Weapon& getCurrentWeapon();
27
28
            /* Cambia el arma actual por la especificada */
29
            void changeWeapon(std::string weapon);
30
31
            typedef WeaponsList::iterator iterator;
32
            typedef WeaponsList::const_iterator const_iterator;
33
            iterator begin();
34
            iterator end();
35
36
   };
37
38
   #endif
39
```

```
WeaponsFactory.cpp
May 27, 18 21:56
                                                                               Page 1/1
    #include "WeaponsFactory.h"
   #include "WeaponNames.h"
   #include "AirAttack.h"
   #include "Banana.h"
   #include "Bat.h"
   #include "Bazooka.h"
   #include "Dynamite.h"
   #include "GreenGrenade.h"
   #include "HolyGrenade.h"
   #include "Mortar.h"
   #include "RedGrenade.h"
   #include "Teleportation.h"
16
   WeaponsFactory::WeaponsFactory() {}
   WeaponsFactory::~WeaponsFactory() {}
20
   weapon_ptr WeaponsFactory::createWeapon(std::string weapon, int ammo) {
21
        if (weapon == AIR ATTACK NAME)
            return weapon ptr(new AirAttack(ammo));
        else if (weapon == BANANA NAME)
23
            return weapon_ptr(new Banana(ammo));
24
25
        else if (weapon == BAT NAME)
            return weapon_ptr(new Bat(ammo));
26
        else if (weapon == BAZOOKA_NAME)
27
            return weapon ptr(new Bazooka(ammo));
28
        else if (weapon == DYNAMITE_NAME)
29
            return weapon_ptr(new Dynamite(ammo));
30
        else if (weapon == GREEN_GRENADE_NAME)
31
32
            return weapon_ptr(new GreenGrenade(ammo));
33
        else if (weapon == HOLY_GRENADE_NAME)
            return weapon_ptr(new HolyGrenade(ammo));
34
        else if (weapon == MORTAR_NAME)
35
36
            return weapon_ptr(new Mortar(ammo));
37
        else if (weapon == RED_GRENADE_NAME)
            return weapon_ptr(new RedGrenade(ammo));
38
        return weapon_ptr(new Teleportation(ammo));
39
40
```

```
WeaponsFactory.h
May 28, 18 18:21
                                                                             Page 1/1
    #ifndef __CLIENTWEAPONSFACTORY_H__
   #define CLIENTWEAPONSFACTORY H
    #include <memorv>
    #include "Weapon.h"
    typedef std::unique ptr<Weapon> weapon ptr;
    /* Clase que se encarga de crear las armas del juego */
   class WeaponsFactory {
        public:
12
            /* Constructor */
13
            WeaponsFactory();
14
15
            /* Destructor */
16
            ~WeaponsFactorv();
17
18
19
            /* Crea el arma especificada con las municiones especificadas */
20
            weapon_ptr createWeapon(std::string weapon, int ammo);
21
22
23
24
   #endif
```

```
Table of Content
Jun 06, 18 22:31
                                                                     Page 1/2
   Table of Contents
    1 ExplosionView.cpp... sheets
                                1 to 1 (1) pages
                                                      1- 1
                                                             44 lines
    2 ExplosionView.h.... sheets 1 to 1 (1) pages
                                                      2- 2
                                                             39 lines
    3 ExplosionViewList.cpp sheets
                                 2 t.o
                                       2 (1) pages
                                                      3-
                                                          3
                                                             24 lines
    4 ExplosionViewList.h. sheets
                                 2 t.o
                                        2 (1) pages
                                                              31 lines
    5 WalkingAnimation.cpp sheets
                                  3 to
                                        3 (1) pages
                                                      5-
                                                             44 lines
                                                      6-
                                                             45 lines
    6 WalkingAnimation.h.. sheets
                                  3 to
                                        3 (1) pages
                                                             80 lines
    7 WeaponAnimation.cpp. sheets
                                  4 to
                                        4 ( 1) pages
    8 WeaponAnimation.h... sheets
                                  5 to
                                        5 (1) pages
                                                             56 lines
    9 main.cpp..... sheets
                                  5 to
                                        5 ( 1) pages
                                                     10- 10
   10 ButtonBuilder.cpp... sheets
                                 6 to
                                        6 (1) pages
                                                     11- 11
                                                             12 lines
                                 6 to
   11 ButtonBuilder.h.... sheets
                                        6 (1) pages
                                                     12- 12
                                                             14 lines
   12 CreateGameMenu.cpp.. sheets
                                 7 to
                                        7 (1) pages
                                                     13- 13
                                                              60 lines
   13 CreateGameMenu.h... sheets
                                 7 to
                                        7 (1) pages
                                                     14- 14
                                                              29 lines
   14 GameMenu.cpp..... sheets
                                  8 to
                                        8 (1) pages
                                                     15- 16
                                                              76 lines
   15 GameMenuField.cpp... sheets
                                  9 to
                                        9 (1) pages
                                                     17- 17
                                                              30 lines
                                 9 to
   16 GameMenuField.h.... sheets
                                        9 ( 1) pages
                                                     18- 18
                                                              35 lines
   17 GameMenu.h...... sheets 10 to 10 (1) pages
                                                     19- 19
                                                             35 lines
   18 JoinGameMenu.cpp.... sheets 10 to 10 (1) pages
                                                     20- 20
                                                             45 lines
   19 JoinGameMenu.h..... sheets 11 to 11 (1) pages
   20 MenuView.cpp...... sheets 11 to 11 (1) pages
   21 MenuView.h..... sheets 12 to 12 (1) pages 23-23
   22 SelectableListMenu.cpp sheets 12 to 12 (1) pages 24-24
   23 SelectableListMenu.h sheets 13 to 13 (1) pages 25-25
                                                              48 lines
   24 ServerFatalError.cpp sheets 13 to 13 (1) pages
                                                     26- 26
                                                              11 lines
   25 ServerFatalError.h.. sheets 14 to 14 (1) pages
                                                     27 - 27
                                                             14 lines
   26 ServerMenu.cpp..... sheets 14 to 15 (2) pages
                                                     28- 29
                                                             65 lines
   27 ServerMenu.h..... sheets 15 to 15 (1) pages
                                                     30- 30
                                                             50 lines
   28 WaitingLabel.cpp.... sheets 16 to 16 (1) pages
                                                     31- 31
                                                             17 lines
   29 WaitingLabel.h..... sheets 16 to 16 (1) pages
                                                     32- 32
   30 MusicPath.h...... sheets 17 to 17 (1) pages 33-33
   31 MusicPlayer.cpp..... sheets 17 to 18 (2) pages 34-36 139 lines
   32 MusicPlayerException.cpp sheets 19 to 19 (1) pages 37-37 13 lines
   33 MusicPlayerException.h sheets 19 to 19 (1) pages 38-38 23 lines
   34 MusicPlayer.h..... sheets 20 to 20 (1) pages 39-40 70 lines
   35 Player.cpp...... sheets 21 to 21 (1) pages 41-42
   36 Player.h..... sheets 22 to 22 (1) pages 43-44
   37 Scope.cpp...... sheets 23 to 23 (1) pages
                                                    45- 45
                                                             33 lines
   38 Scope.h..... sheets 23 to 23 (1) pages
                                                     46- 46
                                                             33 lines
   39 ViewsList.cpp..... sheets 24 to 25 (2) pages
                                                     47- 49
                                                            176 lines
   40 ViewsList.h..... sheets 25 to 26 (2) pages
   41 WeaponPowerAccum.cpp sheets 26 to 26 (1) pages
                                                     52- 52
   42 WeaponPowerAccum.h.. sheets 27 to 27 (1) pages
                                                     53-53
                                                             36 lines
   43 ClientProtocol.cpp.. sheets 27 to 28 (2) pages
                                                     54- 55
                                                            125 lines
   44 ClientProtocol.h... sheets 28 to 29 (2) pages
                                                     56- 57
                                                             74 lines
   45 DataReceiver.cpp.... sheets 29 to
                                      30 (2) pages
                                                     58- 59
                                                             97 lines
   46 DataReceiver.h.... sheets 30 to
                                      30 (1) pages
                                                     60- 60
                                                             34 lines
   47 Handlers.cpp...... sheets 31 to 32 (2) pages
                                                            165 lines
                                                     61- 63
   48 Handlers.h..... sheets 32 to 33 (2) pages
                                                             77 lines
                                                     64- 65
   49 PlayerLifeLabel.cpp. sheets 33 to 33 (1) pages
                                                     66- 66
                                                              40 lines
   50 PlayerLifeLabel.h... sheets 34 to 34 (1) pages
   51 PlayersList.cpp..... sheets 34 to 34 (1) pages
                                                     68- 68
                                                              41 lines
   52 PlayersList.h..... sheets 35 to 35 (1) pages
                                                     69- 69
                                                              46 lines
   53 ScreenView.cpp..... sheets 35 to 35 (1) pages
                                                     70- 70
                                                             64 lines
   54 ScreenView.h...... sheets 36 to 36 (1) pages
                                                     71 - 71
                                                              62 lines
   55 TurnLabel.cpp..... sheets 36 to
                                       36 (1) pages
                                                     72- 72
                                                              51 lines
   56 TurnLabel.h..... sheets 37 to
                                       37 (1) pages
                                                     73- 73
                                                              45 lines
   57 WeaponButton.cpp.... sheets 37 to 37 (1) pages
                                                     74- 74
                                                              40 lines
   58 WeaponButton.h..... sheets 38 to 38 (1) pages
                                                     75 - 75
                                                              38 lines
   59 WeaponView.cpp..... sheets 38 to 38 (1) pages
                                                     76- 76
                                                             30 lines
   60 WeaponView.h..... sheets 39 to 39 (1) pages
                                                     77- 77
                                                     78- 78
   61 WindView.cpp...... sheets 39 to 39 (1) pages
                                                             26 lines
   62 WindView.h..... sheets 40 to 40 (1) pages
                                                     79- 79
   63 WorldView.cpp..... sheets 40 to 41 (2) pages
                                                     80- 81
                                                             96 lines
   64 WorldView.h..... sheets 41 to 42 (2) pages
                                                     82- 83
                                                             67 lines
   65 Turn.cpp...... sheets 42 to 42 (1) pages
                                                     84-84
```

Jun	n 06, 18 22:31	Table	of Content	Page 2/2
67	66 Turn.hs		· , I 3	39 lines
68 69	67 BulletView.cpp s 68 BulletView.h s		· , I 3	31 lines 37 lines
70	69 GirderView.cpp s		44 (1) pages 88-88	27 lines
71	70 GirderView.h s		45 (1) pages 89-89	32 lines
72	71 Viewable.cpp s			28 lines
73 74	72 Viewable.h s 73 WormLifeView.cpp s			46 lines 25 lines
75	74 WormLifeView.hs			32 lines
76	75 WormView.cpps			98 lines
77	76 WormView.h s			97 lines
78	77 DistanceWeapon.cpp s			17 lines
79 80	78 DistanceWeapon.h s 79 MeleeWeapon.cpp s			24 lines 12 lines
81	80 MeleeWeapon.h s			20 lines
82	81 AirAttack.cpp s	heets 51 to	51 (1) pages 102-102	10 lines
83	82 AirAttack.hs			20 lines
84 85	83 Banana.cpp s 84 Banana.h s			10 lines 20 lines
86	85 Bat.cpps			10 lines
87	86 Bat.hs			20 lines
88	87 Bazooka.cpp s			10 lines
89	88 Bazooka.hs			20 lines
90 91	89 Dynamite.cpp s 90 Dynamite.h s			10 lines 20 lines
92	91 GreenGrenade.cpp s			11 lines
93	92 GreenGrenade.h s	heets 57 to	57 (1) pages 113-113	20 lines
94	93 HolyGrenade.cpp s			11 lines
95	94 HolyGrenade.h s			20 lines
96 97	95 Mortar.cpp s 96 Mortar.h s			10 lines 20 lines
98	97 RedGrenade.cpp s			11 lines
99	98 RedGrenade.h s	heets 60 to	60 (1) pages 119-119	20 lines
100	99 Teleportation.cpp s			10 lines
101 102	100 Teleportation.h 101 SelfDirectedWeapon.cp			20 lines 13 lines
102	102 SelfDirectedWeapon.h			23 lines
104	103 Weapon.cpp		62 (1) pages 124-124	56 lines
105	104 Weapon.h			53 lines
106 107	105 WeaponList.cpp			29 lines 40 lines
107	106 WeaponList.h			41 lines
109	108 WeaponsFactory.h			25 lines