

jun 26, 18 2:39

AerialAttack.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#include "AerialAttack.h"

Worm::AerialAttack::AerialAttack(const GUI::GameTextureManager &tex)
    : Weapon(tex, GUI::GameTextures::WormAirAttack, AERIAL_ATTACK_CENTER_FRAME,
WeaponID::WAerial) {
    this->weaponAnimation.setAnimateOnce();
}

void Worm::AerialAttack::update(float dt) {
    if (!this->weaponAnimation.finished()) {
        this->weaponAnimation.update(dt);
    } else {
        this->endAnimation();
    }
}

void Worm::AerialAttack::render(GUI::Position &p, GUI::Camera &cam, SDL_Renderer
Flip &flip) {
    this->weaponAnimation.render(p, cam, flip);
}

void Worm::AerialAttack::setAngle(float angle, Worm::Direction d) {}

void Worm::AerialAttack::startShot() {}

void Worm::AerialAttack::endShot() {}

bool Worm::AerialAttack::positionSelected() {
    this->weaponAnimation.setAutoUpdate(true);
    return true;
}

void Worm::AerialAttack::endAnimation() {
    this->weaponAnimation.setFrame(AERIAL_ATTACK_CENTER_FRAME);
    this->weaponAnimation.setAutoUpdate(false);
}

```

jun 26, 18 2:39

AerialAttack.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#ifndef __AerialAttack_H__
#define __AerialAttack_H__

#define AERIAL_ATTACK_CENTER_FRAME 0

#include "Weapon.h"

namespace Worm {
class AerialAttack : public Weapon {
public:
    explicit AerialAttack(const GUI::GameTextureManager &textureManager);
    ~AerialAttack() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    void endAnimation();
};
} // namespace Worm

#endif //__AerialAttack_H__

```

jun 26, 18 7:40

Armory.cpp

Page 1/2

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 10/06/18
 */

#include <sstream>

#include "Armory.h"

GUI::Armory::Armory(const GUI::GameTextureManager &textureManager, GUI::Camera &cam,
                    GUI::Font &font)
    : manager(textureManager),
      camera(cam),
      font(font),
      weaponButton(font),
      ammunition(WEAPONS_QUANTITY, 0) {}

void GUI::Armory::render() {
    const Texture *temp = this->weaponIcons.back();
    ScreenPosition ammoPos{-temp->getWidth() / 2, 10};
    ScreenPosition iconPos{-temp->getWidth() / 2, 20 + temp->getHeight() / 2};
    ScreenPosition textPos{-temp->getWidth() / 2, 20 + temp->getHeight() * 3 / 2};
};

    int i = 1;
    for (auto &weapon : this->weaponIcons) {
        ammoPos.x += weapon->getWidth();
        iconPos.x += weapon->getWidth();
        textPos.x += weapon->getWidth();

        std::int16_t weaponAmmo = this->ammunition[i - 1];
        std::ostringstream button;
        button << BUTTON_ROOT_STR << i++;

        if (weaponAmmo == -1) {
            weaponButton.set(std::string("inf"), SDL_Color{0, 0, 0}, 20);
            weaponButton.renderFixed(ammoPos, this->camera);
        } else {
            weaponButton.set(std::to_string(weaponAmmo), SDL_Color{0, 0, 0}, 20);
        }

        weaponButton.renderFixed(ammoPos, this->camera);

        weaponButton.set(button.str(), SDL_Color{0, 0, 0}, 25);
        weaponButton.renderFixed(textPos, this->camera);
        this->camera.drawLocal(*weapon, iconPos);
    }

void GUI::Armory::loadWeapons() {
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::BazookaIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::GrenadeIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::ClusterIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::MortarIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::BananaIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::HolyIcon));
}

```

jun 26, 18 7:40

Armory.cpp

Page 2/2

```

        this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::AirIcon));
    });
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::DynamiteIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::BaseballBatIcon));
    this->weaponIcons.emplace_back(&this->manager.get(GUI::GameTextures::TeleportIcon));
}

void GUI::Armory::update(IO::GameStateMsg &msg) {
    for (int i = 0; i < WEAPONS_QUANTITY; i++) {
        this->ammunition[i] = msg.weaponAmmunition[i];
    }
}

```

jun 26, 18 7:40

Armory.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 10/06/18
 */

#ifndef __ARMORY_H__
#define __ARMORY_H__

#define BUTTON_ROOT_STR "F"

#include <vector>

#include <Animation.h>
#include <Font.h>
#include <GameStateMsg.h>
#include <Text.h>
#include "GameTextures.h"

namespace GUI {
class Armory {
public:
    Armory(const GameTextureManager &textureManager, Camera &cam, Font &font);
    ~Armory() = default;
    void loadWeapons();
    void render();
    void update(IO::GameStateMsg &msg);

private:
    const GameTextureManager &manager;
    Camera &camera;
    std::vector<const Texture *> weaponIcons;
    const Font &font;
    Text weaponButton;
    std::vector<std::int16_t> ammunition;
};
} // namespace GUI

#endif //__ARMORY_H__

```

jun 26, 18 2:39	BackFlip.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 20/05/18 */ #include "BackFlip.h" Worm::BackFlip::BackFlip() : State(StateID::StartBackFlip) {} Worm::BackFlip::~BackFlip() {} void Worm::BackFlip::update(float dt) {} IO::PlayerInput Worm::BackFlip::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	BackFlip.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlip::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

BackFlip.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 20/05/18
 */

#ifndef __WORM_BACK_FLIP_H__
#define __WORM_BACK_FLIP_H__

#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class BackFlip : public State {
public:
    explicit BackFlip();
    virtual ~BackFlip();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __WORM_BACK_FLIP_H__

```

jun 29, 18 16:28

BackgroundMusic.cpp

Page 1/1

```
//  
// Created by rodrigo on 25/06/18.  
//  
  
#include "BackgroundMusic.h"  
#include "Exception.h"  
  
GUI::BackgroundMusic::BackgroundMusic(const std::string &filename) {  
    this->backgroundMusic = Mix_LoadMUS(filename.c_str());  
    if (!this->backgroundMusic) {  
        throw Exception{"Error loading %s: %s", filename.c_str(), Mix_GetError()};  
    }  
}  
  
GUI::BackgroundMusic::~BackgroundMusic() {  
    if (this->backgroundMusic != nullptr) {  
        Mix_FreeMusic(this->backgroundMusic);  
    }  
}  
  
Mix_Music *GUI::BackgroundMusic::getMusic() const {  
    return this->backgroundMusic;  
}  
  
GUI::BackgroundMusic::BackgroundMusic(GUI::BackgroundMusic &&other) {  
    std::swap(this->backgroundMusic, other.backgroundMusic);  
}  
  
void GUI::BackgroundMusic::play() const {  
    Mix_PlayMusic(this->backgroundMusic, -1);  
}
```

jun 29, 18 16:28

BackgroundMusic.h

Page 1/1

```
//  
// Created by rodrigo on 25/06/18.  
//  
  
#ifndef INC_4_WORMS_BACKGROUNDDMUSIC_H  
#define INC_4_WORMS_BACKGROUNDDMUSIC_H  
  
#include <SDL2/SDL.h>  
#include <SDL2/SDL_mixer.h>  
#include <string>  
  
namespace GUI {  
    class BackgroundMusic {  
    public:  
        BackgroundMusic(const std::string &filename);  
        BackgroundMusic(BackgroundMusic &&other);  
        ~BackgroundMusic();  
        Mix_Music *getMusic() const;  
        void play() const;  
  
    private:  
        Mix_Music *backgroundMusic{nullptr};  
    };  
}  
  
#endif //INC_4_WORMS_BACKGROUNDDMUSIC_H
```


jun 29, 18 16:28

BackgroundMusicManager.h

Page 1/1

```

//
// Created by rodrigo on 25/06/18.
//

#ifndef INC_4_WORMS_BACKGROUNDDMUSICMANAGER_H
#define INC_4_WORMS_BACKGROUNDDMUSICMANAGER_H

#include <SDL2/SDL.h>
#include <functional>
#include <string>
#include <unordered_map>
#include "BackgroundMusic.h"

namespace GUI {
    template <typename ID, typename HASH = std::hash<ID>>
    class BackgroundMusicManager {
    public:
        BackgroundMusicManager();
        ~BackgroundMusicManager();
        BackgroundMusicManager& operator=(BackgroundMusicManager& other) = delet
e;

        void load(ID id, const std::string& file_name);
        const BackgroundMusic& get(ID id) const;

    private:
        std::unordered_map<ID, BackgroundMusic, HASH> cache;
    };
} // namespace GUI

template <typename ID, typename HASH>
GUI::BackgroundMusicManager<ID, HASH>::BackgroundMusicManager() {}

template <typename ID, typename HASH>
GUI::BackgroundMusicManager<ID, HASH>::~~BackgroundMusicManager() {}

/**
 * @brief Loads a background music file.
 *
 * @param file_name The image file name.
 */
template <typename ID, typename HASH>
void GUI::BackgroundMusicManager<ID, HASH>::load(ID id, const std::string& file_
name) {
    GUI::BackgroundMusic backgroundMusic{file_name};
    this->cache.insert(std::make_pair(id, std::move(backgroundMusic)));
}

/**
 * @brief Gets a background music.
 *
 * @param file_name Name of the background music.
 */
template <typename ID, typename HASH>
const GUI::BackgroundMusic& GUI::BackgroundMusicManager<ID, HASH>::get(ID id) co
nst {
    return this->cache.at(id);
}

#endif //INC_4_WORMS_BACKGROUNDDMUSICMANAGER_H

```

jun 29, 18 16:28

BackgroundMusicPlayer.cpp

Page 1/1

```
//  
// Created by rodrigo on 25/06/18.  
//  
  
#include "BackgroundMusicPlayer.h"  
  
GUI::BackgroundMusicPlayer::BackgroundMusicPlayer(const GUI::BackgroundMusic &backgroundMusic)  
    : backgroundMusic(&backgroundMusic) {}  
  
GUI::BackgroundMusicPlayer::~BackgroundMusicPlayer() {}  
  
void GUI::BackgroundMusicPlayer::play() {  
    this->backgroundMusic->play();  
}
```

jun 29, 18 16:28

BackgroundMusicPlayer.h

Page 1/1

```
//  
// Created by rodrigo on 25/06/18.  
//  
  
#ifndef INC_4_WORMS_BACKGROUNDDMUSICPLAYER_H  
#define INC_4_WORMS_BACKGROUNDDMUSICPLAYER_H  
  
#include <SDL2/SDL.h>  
  
#include "BackgroundMusic.h"  
  
namespace GUI {  
    class BackgroundMusicPlayer {  
    public:  
        bool loop{false};  
  
        explicit BackgroundMusicPlayer(const GUI::BackgroundMusic &backgroundMus  
ic);  
        ~BackgroundMusicPlayer();  
        void play();  
  
    private:  
        const BackgroundMusic *backgroundMusic;  
    };  
}  
  
#endif //INC_4_WORMS_BACKGROUNDDMUSICPLAYER_H
```

jun 26, 18 2:39

Banana.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include <cmath>

#include "Banana.h"

Worm::Banana::Banana(const GUI::GameTextureManager &tex)
: Weapon(tex, GUI::GameTextures::WormBanana, BANANA_CENTER_FRAME, WeaponID::
WBanana),
  scope(this->textureMgr),
  powerBar(this->textureMgr) {}

void Worm::Banana::update(float dt) {
    this->weaponAnimation.update(dt);
    this->scope.update(dt);
    this->powerBar.update(dt);
}

void Worm::Banana::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &
flip) {
    this->weaponAnimation.render(p, cam, flip);
    this->scope.render(p, cam, flip);
    this->powerBar.render(p, cam, flip);
}

void Worm::Banana::setAngle(float angle, Worm::Direction d) {
    this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->ce
nterFrame);
    this->scope.setAngle(angle, d);
    this->powerBar.setAngle(angle, d);
}

void Worm::Banana::startShot() {
    this->powerBar.startShot();
}

void Worm::Banana::endShot() {
    this->powerBar.endShot();
}

bool Worm::Banana::positionSelected() {
    return false;
}

```

jun 26, 18 2:39

Banana.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __BANANA_H__
#define __BANANA_H__

#include <vector>

#include "PowerBar.h"
#include "Scope.h"
#include "Weapon.h"

#define BANANA_CENTER_FRAME 14

namespace Worm {
class Banana : public Weapon {
public:
    explicit Banana(const GUI::GameTextureManager &textureManager);
    ~Banana() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
    ::Weapon::PowerBar powerBar;
};
} // namespace Worm

#endif //__BANANA_H__

```

jun 26, 18 2:39

BaseballBat.cpp

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#include "BaseballBat.h"
#include <cmath>
#include <iostream>

Worm::BaseballBat::BaseballBat(const GUI::GameTextureManager &tex)
    : Weapon(tex, GUI::GameTextures::WormBaseballBat, BASEBALL_BAT_CENTER_FRAME,
             WeaponID::WBaseballBat),
      scope(this->textureMgr) {}

void Worm::BaseballBat::update(float dt) {
    this->weaponAnimation.update(dt);
    this->scope.update(dt);
}

void Worm::BaseballBat::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererF
lip &flip) {
    this->weaponAnimation.render(p, cam, flip);
    this->scope.render(p, cam, flip);
}

void Worm::BaseballBat::setAngle(float angle, Direction d) {
    this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->ce
nterFrame);
    this->scope.setAngle(angle, d);
}

void Worm::BaseballBat::startShot() {}

void Worm::BaseballBat::endShot() {}

bool Worm::BaseballBat::positionSelected() {
    return false;
}

```

jun 26, 18 2:39

BaseballBat.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_BASEBALLBAT_H
#define INC_4_WORMS_BASEBALLBAT_H

#include "Scope.h"
#include "Weapon.h"

#define BASEBALL_BAT_CENTER_FRAME 16

namespace Worm {
class BaseballBat : public Weapon {
public:
    explicit BaseballBat(const GUI::GameTextureManager &textureManager);
    ~BaseballBat() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
};
} // namespace Worm

#endif // INC_4_WORMS_BASEBALLBAT_H

```

jun 26, 18 2:39	Batting.cpp	Page 1/2
<pre>// // Created by rodrigo on 23/06/18. // #include "Batting.h" Worm::Batting::Batting() : State(StateID::Batting) {} Worm::Batting::~Batting() {} void Worm::Batting::update(float dt) {} IO::PlayerInput Worm::Batting::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::mortar(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39	Batting.cpp	Page 2/2
<pre>} IO::PlayerInput Worm::Batting::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Batting::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 7:40	Batting.h	Page 1/2
<pre>// // Created by rodrigo on 23/06/18. // #ifndef INC_4_WORMS_BATTING_H #define INC_4_WORMS_BATTING_H #include "../Worm.h" #include "GameStateMsg.h" #include "WormState.h" namespace Worm { class Batting : public State { public: Batting(); ~Batting(); virtual void update(float dt) override; virtual IO::PlayerInput moveRight(Worm &w) override; virtual IO::PlayerInput moveLeft(Worm &w) override; virtual IO::PlayerInput stopMove(Worm &w) override; virtual IO::PlayerInput jump(Worm &w) override; virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override; virtual IO::PlayerInput bazooka(Worm &w) override; virtual IO::PlayerInput grenade(Worm &w) override; virtual IO::PlayerInput cluster(Worm &w) override; virtual IO::PlayerInput mortar(Worm &w) override; virtual IO::PlayerInput banana(Worm &w) override; virtual IO::PlayerInput holy(Worm &w) override; virtual IO::PlayerInput aerialAttack(Worm &w) override; virtual IO::PlayerInput dynamite(Worm &w) override; virtual IO::PlayerInput baseballBat(Worm &w) override; virtual IO::PlayerInput teleport(Worm &w) override; virtual IO::PlayerInput positionSelected(Worm &w) override; virtual IO::PlayerInput startShot(Worm &w) override; virtual IO::PlayerInput endShot(Worm &w) override; virtual IO::PlayerInput pointUp(Worm &w) override; virtual IO::PlayerInput pointDown(Worm &w) override; virtual IO::PlayerInput backFlip(Worm &w) override; };</pre>		

jun 26, 18 7:40	Batting.h	Page 2/2
<pre>} #endif // INC_4_WORMS_BATTING_H</pre>		

jun 26, 18 2:39

Bazooka.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include <cmath>

#include "Bazooka.h"

Worm::Bazooka::Bazooka(const GUI::GameTextureManager &tex)
    : Weapon(tex, GUI::GameTextures::Bazooka, BAZOOKA_CENTER_FRAME, WeaponID::WBazooka),
      scope(this->textureMgr),
      powerBar(this->textureMgr) {}

void Worm::Bazooka::update(float dt) {
    this->weaponAnimation.update(dt);
    this->scope.update(dt);
    this->powerBar.update(dt);
}

void Worm::Bazooka::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) {
    this->weaponAnimation.render(p, cam, flip);
    this->scope.render(p, cam, flip);
    this->powerBar.render(p, cam, flip);
}

void Worm::Bazooka::setAngle(float angle, Direction d) {
    this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->centerFrame);
    this->scope.setAngle(angle, d);
    this->powerBar.setAngle(angle, d);
}

void Worm::Bazooka::startShot() {
    this->powerBar.startShot();
}

void Worm::Bazooka::endShot() {
    this->powerBar.endShot();
}

bool Worm::Bazooka::positionSelected() {
    return false;
}

```

jun 26, 18 2:39

Bazooka.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __BAZOOKA_H__
#define __BAZOOKA_H__

#include <vector>

#include "PowerBar.h"
#include "Scope.h"
#include "Weapon.h"

#define BAZOOKA_CENTER_FRAME 16

namespace Worm {
class Bazooka : public Weapon {
public:
    explicit Bazooka(const GUI::GameTextureManager &textureManager);
    ~Bazooka() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
    ::Weapon::PowerBar powerBar;
};
} // namespace Worm

#endif //__BAZOOKA_H__

```

jun 26, 18 2:39	Bullet.cpp	Page 1/3
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 26/05/18 */ #include <cmath> #include <iostream> #include "../GameSoundEffects.h" #include "Bullet.h" Ammo::Bullet::Bullet(const GUI::GameTextureManager &texture_mgr, const GUI::GameSoundEffectManager &sound_effect_mgr, Worm:: WeaponID id) : texture_mgr(texture_mgr), sound_effect_mgr(sound_effect_mgr), animation(this->texture_mgr.get(GUI::GameTextures::Missile), true, MISSILE _0_DEG_FRAME, false), explosion(this->texture_mgr) { switch (id) { case Worm::WeaponID::WBazooka: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::Missile), true, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WGrenade: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::Grenade), false, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WCluster: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::Cluster), false, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WMortar: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::Mortar), false, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WBanana: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::Banana), false, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WHoly: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::Holy), false, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WAerial: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameText ures::AirMissile), false, MISSILE_0_DEG_FRAME, false); break; case Worm::WeaponID::WBaseballBat: break; case Worm::WeaponID::WTeleport: break; case Worm::WeaponID::WExplode: break; } </pre>		

jun 26, 18 2:39	Bullet.cpp	Page 2/3
<pre> case Worm::WeaponID::WFragment: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameTextures::Fragment), false, 0, true); this->updateManually = false; break; case Worm::WeaponID::WDynamite: this->animation = GUI::Animation(this->texture_mgr.get(GUI::GameTextures::Dynamite), false, 0, true); this->updateManually = false; break; case Worm::WeaponID::WNone: break; } this->wid = id; } void Ammo::Bullet::update(float dt) { if (!this->explode) { if (this->updateManually) { float angle = (this->angle - 90); if (angle >= 360) { angle -= 360; } float angleStep = MISSILE_ANGLE_STEP; this->animation.setFrame((int)std::floor(angle / angleStep)); } else { this->animation.update(dt); } } else { this->explosion.update(dt); } } void Ammo::Bullet::render(GUI::Position p, GUI::Camera &cam) { if (!this->explode) { this->animation.render(p, cam, SDL_FLIP_HORIZONTAL); } else { this->explosion.render(cam); } } void Ammo::Bullet::setAngle(float angle) { this->angle = angle; } bool Ammo::Bullet::exploded() { return this->explosion.finished(); } void Ammo::Bullet::madeImpact() { this->explode = true; this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::S oundEffectPlayer{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Explosion), true}); this->soundEffectPlayer->play(); } void Ammo::Bullet::setPosition(GUI::Position p) { this->position = p; this->explosion.position = p; } </pre>		

jun 26, 18 2:39

Bullet.cpp

Page 3/3

```
}  
  
bool Ammo::Bullet::exploding() {  
    return this->explode;  
}  
  
GUI::Position Ammo::Bullet::getPosition() {  
    return this->position;  
}
```

jun 26, 18 2:39

Bullet.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 26/05/18
 */

#ifndef __Bullet_h__
#define __Bullet_h__

#include <GameStateMsg.h>
#include <memory>

#include "../GameSoundEffects.h"
#include "../GameTextures.h"
#include "../SoundEffectPlayer.h"
#include "Animation.h"
#include "Explosion.h"

#define MISSILE_0_DEG_FRAME 8
#define MISSILE_ANGLE_STEP 11.25f

namespace Ammo {
class Bullet {
public:
    explicit Bullet(const GUI::GameTextureManager &texture_mgr,
                   const GUI::GameSoundEffectManager &sound_effect_mgr, Worm::W
eaponID id);
    ~Bullet() = default;
    void update(float dt);
    void render(GUI::Position p, GUI::Camera &cam);
    void setAngle(float angle);
    void setPosition(GUI::Position p);
    GUI::Position getPosition();
    void madeImpact();
    bool exploding();
    bool exploded();

private:
    float angle{0};
    bool updateManually{true};
    const GUI::GameTextureManager &texture_mgr;
    const GUI::GameSoundEffectManager &sound_effect_mgr;
    GUI::Animation animation;
    GUI::Position position{0, 0};
    Worm::Explosion explosion;
    bool explode{false};
    Worm::WeaponID wid;
    std::shared_ptr<GUI::SoundEffectPlayer> soundEffectPlayer{nullptr};
};
}

#endif //__Bullet_H__

```

jun 29, 18 16:28	Button.cpp	Page 1/2
<pre>// // Created by rodrigo on 20/06/18. // #include <Font.h> #include "Button.h" GUI::Button::Button(ScreenPosition sp, int height, int width, const std::string &msg, Font &font) : position(sp), height(height), width(width), msg(msg), textColor(SDL_Color{0xFF, 0xFF, 0xFF}), textSize(40), text(font) { this->text.set(this->msg, SDL_Color{0xFF, 0xFF, 0xFF}, 40); } GUI::Button::Button(const std::string &msg, GUI::Font &font, SDL_Color textColor , int textSize) : msg(msg), textColor(textColor), textSize(textSize), text(font) { this->text.set(this->msg, textColor, textSize); } GUI::Button::Button(const std::string &msg, GUI::Font &font) : msg(msg), text(font) { } GUI::Button::Button(GUI::ScreenPosition sp, int height, int width, GUI::Font &fo nt) : position(sp), height(height), width(width), text(font) { } void GUI::Button::render(GUI::Camera &cam) { SDL_Rect fillRect = {this->position.x - this->width / 2, this->position.y + this->height / 2, this->width / (int) cam.getScale(), this->height / (int) cam.getScale()}; cam.drawLocal(ScreenPosition{this->position.x, this->position.y}, fillRect, SDL_Color{0, 0, 0}); this->text.set(this->msg, this->textColor, this->textSize); this->text.renderFixed(this->position, cam); } bool GUI::Button::inside(GUI::ScreenPosition sp) { bool inside = true; if(sp.x < this->position.x - this->width / 2) { //Mouse is left of the button inside = false; } else if(sp.x > this->position.x + this->width / 2) { //Mouse is right of the button inside = false; } }</pre>		

jun 29, 18 16:28	Button.cpp	Page 2/2
<pre> } else if(sp.y < this->position.y - this->height / 2) { //Mouse below the button inside = false; } else if(sp.y > this->position.y + this->width / 2) { //Mouse above the button inside = false; } return inside; } void GUI::Button::setBackground(SDL_Color color) { this->text.setBackground(color); }</pre>		

jun 29, 18 16:28

Button.h

Page 1/1

```
//  
// Created by rodrigo on 20/06/18.  
//  
  
#ifndef INC_4_WORMS_BUTTON_H  
#define INC_4_WORMS_BUTTON_H  
  
#include <Camera.h>  
#include <Text.h>  
  
namespace GUI {  
    class Button {  
    public:  
        GUI::ScreenPosition position{0, 0};  
        int height{0};  
        int width{0};  
        std::string msg;  
        SDL_Color textColor{0, 0, 0};  
        int textSize{10};  
  
        Button(ScreenPosition sp, int height, int width, const std::string &msg,  
Font &font);  
        Button(const std::string &msg, GUI::Font &font, SDL_Color textColor, int  
textSize);  
        Button(const std::string &msg, Font &font);  
        Button(ScreenPosition sp, int height, int width, Font &font);  
  
        bool inside(ScreenPosition sp);  
        void render(GUI::Camera &cam);  
  
        void setBackground(SDL_Color color);  
  
    private:  
        Text text;  
    };  
}  
  
#endif //INC_4_WORMS_BUTTON_H
```


jun 26, 18 2:39

ClientSocket.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#include <netdb.h>
#include <unistd.h>
#include <cstring>

#include "ClientSocket.h"
#include "ErrorMessages.h"
#include "Exception.h"

ClientSocket::ClientSocket(const char *hostName, const char *port) {
    int status;
    bool is_connected = false;

    struct addrinfo hints = {AI_PASSIVE, AF_INET, SOCK_STREAM, 0, 0, nullptr, nullptr, nullptr, nullptr};
    struct addrinfo *result, *ptr;

    status = getaddrinfo(hostName, port, &hints, &result);
    if (status != 0) {
        throw Exception(ERR_MSG_SOCKET_INVALID_HOST_OR_PORT, hostName, port, strerror(errno));
    }

    for (ptr = result; ptr != nullptr && !is_connected; ptr = ptr->ai_next) {
        this->fd = socket(ptr->ai_family, ptr->ai_socktype, ptr->ai_protocol);
        /*
         * si la creaci3n del socket falla, no debo hacer nada mas
         * en el ciclo (ya que no se abrio ningun fd)
         */
        if (this->fd == -1) {
            continue;
        }

        status = ::connect(this->fd, ptr->ai_addr, ptr->ai_addrlen);
        if (status == -1) {
            ::close(this->fd);
            this->fd = -1;
        } else {
            is_connected = true;
        }
    }

    freeaddrinfo(result);
    if (!is_connected) {
        throw Exception(ERR_MSG_CONNECTION_COULD_NOT_BE_STABLISHED, hostName, port);
    }
}

```

jun 26, 18 2:39

ClientSocket.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#ifndef __ClientSocket_H__
#define __ClientSocket_H__

#include <string>

#include "CommunicationSocket.h"

/**
 * Socket que tiene la capacidad de realizar una conexion con el servidor,
 * partiendo del dato del host y el port a donde conectarse
 */
class ClientSocket : public CommunicationSocket {
public:
    ClientSocket(const char *hostName, const char *port);
};

#endif //__ClientSocket_H__
```

jun 26, 18 2:39

Cluster.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include <cmath>

#include "Cluster.h"

Worm::Cluster::Cluster(const GUI::GameTextureManager &tex)
: Weapon(tex, GUI::GameTextures::WormCluster, CLUSTER_CENTER_FRAME, WeaponID
::WCluster),
  scope(this->textureMgr),
  powerBar(this->textureMgr) {}

void Worm::Cluster::update(float dt) {
    this->weaponAnimation.update(dt);
    this->scope.update(dt);
    this->powerBar.update(dt);
}

void Worm::Cluster::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip
&flip) {
    this->weaponAnimation.render(p, cam, flip);
    this->scope.render(p, cam, flip);
    this->powerBar.render(p, cam, flip);
}

void Worm::Cluster::setAngle(float angle, Worm::Direction d) {
    this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->ce
nterFrame);
    this->scope.setAngle(angle, d);
    this->powerBar.setAngle(angle, d);
}

void Worm::Cluster::startShot() {
    this->powerBar.startShot();
}

void Worm::Cluster::endShot() {
    this->powerBar.endShot();
}

bool Worm::Cluster::positionSelected() {
    return false;
}

```

jun 26, 18 2:39

Cluster.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __CLUSTER_H__
#define __CLUSTER_H__

#include <vector>

#include "PowerBar.h"
#include "Scope.h"
#include "Weapon.h"

#define CLUSTER_CENTER_FRAME 15

namespace Worm {
class Cluster : public Weapon {
public:
    explicit Cluster(const GUI::GameTextureManager &textureManager);
    ~Cluster() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
    ::Weapon::PowerBar powerBar;
};
} // namespace Worm

#endif //__CLUSTER_H__

```

jun 29, 18 16:28

CommunicationProtocol.cpp

Page 1/3

```
//
// Created by rodrigo on 20/06/18.
//

#include <fstream>
#include <iostream>

#include "CommunicationProtocol.h"
#include "GameStateMsg.h"
#include "Stream.h"

IO::CommunicationProtocol::CommunicationProtocol(ClientSocket &socket, IO::Stream<IO::ClientGUIMsg> *clientStream,
                                                IO::Stream<IO::ServerResponse>
*output)
:
protocol(socket),
clientStream(clientStream),
output(output) {}

void IO::CommunicationProtocol::run() {
    try {
        while (!this->quit) {
            IO::ClientGUIMsg msg;
            if (clientStream->pop(msg)) {
                this->handleClientInput(msg);
            }
        }
        catch (std::exception &e){
            if (!this->quit){
                std::cerr << "In CommunicationProtocol::run()" << std::endl;
                std::cerr << e.what() << std::endl;
                IO::ServerResponse sr{IO::ServerResponseAction::serverClosed};
                *this->output << sr;
            }
        }
        catch (...) {
            std::cerr << "Unknown Error in CommunicationProtocol::run()" << std::endl;
        }
    }

void IO::CommunicationProtocol::startCreateGame() {
    this->command = COMMAND_GET_LEVELS;
    this->protocol << this->command;
    this->protocol >> this->levelsInfo;
    *this->output << IO::ServerResponse{IO::ServerResponseAction::levelsInfo};
}

void IO::CommunicationProtocol::createGame() {
    this->command = COMMAND_CREATE_GAME;
    this->protocol << this->command;
    this->protocol << this->levelToCreate;
    this->getLevelFiles();
    this->waitGameStart(this->levelsInfo[this->levelToCreate].playersQuantity);
}

void IO::CommunicationProtocol::startJoinGame() {
    this->command = COMMAND_GET_GAMES;
    this->protocol << this->command;
    this->protocol >> this->gamesInfo;

    IO::ServerResponse sr;
```

jun 29, 18 16:28

CommunicationProtocol.cpp

Page 2/3

```
sr.action = IO::ServerResponseAction::gamesInfo;
*this->output << sr;
}

void IO::CommunicationProtocol::joinGame() {
    this->command = COMMAND_JOIN_GAME;
    this->protocol << this->command;
    this->protocol << this->gameToJoin;
    this->protocol << this->levelOfGameToJoin;
    this->getLevelFiles();
    this->waitGameStart(this->gamesInfo[this->gameToJoin].numTotalPlayers);
}

ClientSocket IO::CommunicationProtocol::getSocket() {
    return std::move(this->protocol.getSocket());
}

void IO::CommunicationProtocol::waitGameStart(uint8_t playersQuantity) {
    while (this->playersQuantity < playersQuantity) {
        this->protocol >> this->playersQuantity;
        *this->output << IO::ServerResponse{IO::ServerResponseAction::playerConnected};
    }
    IO::ServerResponse sr{};
    sr.action = IO::ServerResponseAction::startGame;
    *this->output << sr;
}

void IO::CommunicationProtocol::stop() {
    this->quit = true;
    this->protocol.stopCommunication();
}

void IO::CommunicationProtocol::handleClientInput(IO::ClientGUIMsg &msg) {
    switch (msg.input) {
        case IO::ClientGUIInput::startCreateGame: {
            this->startCreateGame();
            break;
        }
        case IO::ClientGUIInput::levelSelected: {
            this->createGame();
            break;
        }
        case IO::ClientGUIInput::startJoinGame: {
            this->startJoinGame();
            break;
        }
        case IO::ClientGUIInput::joinGame: {
            this->joinGame();
            break;
        }
        case IO::ClientGUIInput::quit: {
            this->quit = true;
            break;
        }
        default: {
            break;
        }
    }
}

void IO::CommunicationProtocol::getLevelFiles() {
```

jun 29, 18 16:28

CommunicationProtocol.cpp

Page 3/3

```
this->protocol >> this->levelPath;  
std::ofstream levelFile(this->levelPath, std::ofstream::binary);  
this->protocol >> levelFile;  
  
this->protocol >> this->backgroundPath;  
for (auto &background : this->backgroundPath) {  
    std::ofstream backgroundFile(background, std::ofstream::binary);  
    this->protocol >> backgroundFile;  
}  
}
```

jun 29, 18 16:28

CommunicationProtocol.h

Page 1/1

```

//
// Created by rodrigo on 20/06/18.
//

#ifndef INC_4_WORMS_COMMUNICATIONPROTOCOL_H
#define INC_4_WORMS_COMMUNICATIONPROTOCOL_H

#include "ClientSocket.h"
#include <Protocol.h>
#include <Stream.h>
#include "Thread.h"

namespace IO {
    class CommunicationProtocol : public Thread {
    public:
        std::vector<LevelInfo> levelsInfo;
        uint8_t levelToCreate{0};
        std::vector<GameInfo> gamesInfo;
        uint8_t gameToJoin{0};
        uint8_t levelOfGameToJoin{0};
        std::string levelPath;
        std::vector<std::string> backgroundPath;

        explicit CommunicationProtocol(ClientSocket &socket, IO::Stream<IO::ClientGUIMsg> *clientStream,
                                      IO::Stream<IO::ServerResponse> *output);

        void run() override;
        void stop() override;

        ClientSocket getSocket();

    private:
        Protocol<ClientSocket> protocol;
        unsigned char command{0};
        std::uint8_t playersQuantity{0};
        IO::Stream<IO::ClientGUIMsg> *clientStream;
        IO::Stream<IO::ServerResponse> *output;
        bool quit{false};

        void startCreateGame();
        void startJoinGame();
        void joinGame();
        void waitGameStart(uint8_t playersQuantity);

        void handleClientInput(ClientGUIMsg &msg);

        void createGame();

        void getLevelFiles();
    };
}

#endif //INC_4_WORMS_COMMUNICATIONPROTOCOL_H

```

jun 29, 18 16:28

ConnectionWindow.cpp

Page 1/2

```
//
// Created by rodrigo on 24/06/18.
//

#include "ConnectionWindow.h"

GUI::ConnectionWindow::ConnectionWindow(GUI::Window &window, GUI::Font &font, GUI::Camera &cam) :
    GameWindow(window, font, cam) {
    std::string msg(CONNECT_MSG);
    this->buttons.emplace_back(msg, this->font, SDL_Color{0xFF, 0xFF, 0xFF}, this->textSize);
    int x = this->window.getWidth() / 2;
    int y = this->window.getHeight() * 3 / 4;
    this->buttons.back().position = ScreenPosition{x, y};
    this->buttons.back().height = this->textSize * 3 / 2;
    this->buttons.back().width = this->buttons.back().msg.size() * 20 + 20;

    x = this->window.getWidth() * 6 / 10;
    y = this->window.getHeight() * 2 / 7;
    int textFieldHeight = this->textSize * 3 / 2;
    int textFieldWidth = 400;
    std::string emptyMsg("");
    this->textFields.emplace_back(emptyMsg, ScreenPosition{x, y}, textFieldHeight, textFieldWidth, this->font);
    y = this->window.getHeight() * 4 / 7;
    emptyMsg = "";
    this->textFields.emplace_back(emptyMsg, ScreenPosition{x, y}, textFieldHeight, textFieldWidth, this->font);
}

void GUI::ConnectionWindow::start() {
}

void GUI::ConnectionWindow::render() {
    this->window.clear(SDL_Color{0xFF, 0xFF, 0xFF});

    SDL_Color black{0, 0, 0};

    Text ip(this->font);
    Text port(this->font);
    port.setBackground(black);
    int x = this->window.getWidth() * 3 / 10;
    int y = this->window.getHeight() * 2 / 7;
    ip.set("IP:", black, 50);
    ip.renderFixed(ScreenPosition{x, y}, this->cam);
    y = this->window.getHeight() * 4 / 7;
    ip.set("Server port:", black, 50);
    ip.renderFixed(ScreenPosition{x, y}, this->cam);

    for (auto &button : this->buttons) {
        button.render(this->cam);
    }

    for (auto &textField : this->textFields) {
        textField.render(this->cam);
    }

    this->window.render();
}
```

jun 29, 18 16:28

ConnectionWindow.cpp

Page 2/2

```
void GUI::ConnectionWindow::buttonPressed(GUI::ScreenPosition sp) {
    for (auto &textField : this->textFields) {
        textField.selected(sp);
    }

    if (this->buttons[0].inside(sp)) {
        this->notify(*this, Event::ConnectionToServer);
    }
}

void GUI::ConnectionWindow::appendCharacter(char *text) {
    for (auto &textField : this->textFields) {
        if (textField.focus) {
            textField.appendCharacter(text);
        }
    }
}

void GUI::ConnectionWindow::handleKeyDown(SDL_Keycode key) {
    switch (key) {
        case SDLK_BACKSPACE: {
            for (auto &textField : this->textFields) {
                if (textField.focus) {
                    textField.backSpace();
                }
            }
            break;
        }
    }
}

GUI::ConnectionInfo GUI::ConnectionWindow::getConnectionInfo() {
    return ConnectionInfo{this->textFields[0].inputText.msg.c_str(),
                          this->textFields[1].inputText.msg.c_str()};
}
```


jun 29, 18 16:28

ConnectionWindow.h

Page 1/1

```

//
// Created by rodrigo on 24/06/18.
//

#ifndef INC_4_WORMS_CONNECTIONWINDOW_H
#define INC_4_WORMS_CONNECTIONWINDOW_H

#include <vector>

#include "Window.h"
#include "Font.h"
#include "GameStateMsg.h"
#include "GameWindow.h"
#include "Button.h"

#define CONNECT_MSG "Connect"
#define IP_FOCUS 0
#define PORT_FOCUS 1

namespace GUI {
    struct ConnectionInfo {
        const char *ip;
        const char *port;
    };
    class ConnectionWindow : public GameWindow {
    public:
        uint8_t playersConnected{0};

        explicit ConnectionWindow(GUI::Window &window, GUI::Font &font, GUI::Camera &cam);

        void start() override;
        void render() override;
        void handleKeyDown(SDL_Keycode key) override;
        void appendCharacter(char text[32]) override;
        void buttonPressed(ScreenPosition sp) override;

        ConnectionInfo getConnectionInfo();

    private:
        std::vector<Button> buttons;
        int textSize{50};
    };
}

#endif //INC_4_WORMS_CONNECTIONWINDOW_H

```

jun 29, 18 16:28

CreateGameWindow.cpp

Page 1/2

```
//
// Created by rodrigo on 23/06/18.
//

#include <iostream>
#include "GameStateMsg.h"
#include "CreateGameWindow.h"

GUI::CreateGameWindow::CreateGameWindow(GUI::Window &window, GUI::Font &font, GUI::Camera &cam,
                                       std::vector<IO::LevelInfo> &levelsInfo)
:
    GameWindow(window, font, cam),
    levelsInfo(levelsInfo) {
    int height = this->levelInfoSize * 3 / 2;
    std::string msg(SELECT_LEVEL_MSG);
    int x = this->window.getWidth() / 2;
    int y = this->window.getHeight() * 3 / 4;
    this->buttons.emplace_back(msg, this->font, SDL_Color{0xFF, 0xFF, 0xFF}, this->levelInfoSize);
    this->buttons.back().position = ScreenPosition{x, y};
    this->buttons.back().height = height;
    this->buttons.back().width = this->buttons.back().msg.size() * 9 + 20;
    msg = NEXT_LEVEL_MSG;
    x = this->window.getWidth() * 3 / 4;
    y = this->window.getHeight() / 2;
    this->buttons.emplace_back(msg, this->font, SDL_Color{0xFF, 0xFF, 0xFF}, this->levelInfoSize);
    this->buttons.back().position = ScreenPosition{x, y};
    this->buttons.back().height = height;
    this->buttons.back().width = this->buttons.back().msg.size() * 9 + 20;
    msg = PREVIOUS_LEVEL_MSG;
    x = this->window.getWidth() / 4;
    y = this->window.getHeight() / 2;
    this->buttons.emplace_back(msg, this->font, SDL_Color{0xFF, 0xFF, 0xFF}, this->levelInfoSize);
    this->buttons.back().position = ScreenPosition{x, y};
    this->buttons.back().height = height;
    this->buttons.back().width = this->buttons.back().msg.size() * 9 + 20;
}

void GUI::CreateGameWindow::start() {
}

void GUI::CreateGameWindow::render() {
    this->window.clear(SDL_Color{0xFF, 0xFF, 0xFF});

    SDL_Color white{0xFF, 0xFF, 0xFF};
    SDL_Color black{0, 0, 0};

    Text levelName(this->font);
    Text levelPlayersQuantity(this->font);
    levelName.setBackground(black);
    levelPlayersQuantity.setBackground(black);
    int x = this->window.getWidth() * 4 / 10;
    int y = this->window.getHeight() * 3 / 7;
    levelName.set(LEVEL_MSG, white, 50);
    levelName.renderFixed(ScreenPosition{x, y - 50}, this->cam);
    x = this->window.getWidth() * 6 / 10;
    levelName.set(PLAYERS_MSG, white, 50);
    levelName.renderFixed(ScreenPosition{x, y - 50}, this->cam);
```

jun 29, 18 16:28

CreateGameWindow.cpp

Page 2/2

```
levelName.setBackground(white);
levelPlayersQuantity.setBackground(white);
x = this->window.getWidth() * 4 / 10;
y = this->window.getHeight() / 2;
levelName.set(this->levelsInfo[this->buttonSelected].name, black, this->levelInfoSize);
levelName.renderFixed(ScreenPosition{x, y}, this->cam);
x = this->window.getWidth() * 6 / 10;
levelName.set(std::to_string(this->levelsInfo[this->buttonSelected].playersQuantity), black, this->levelInfoSize);
levelName.renderFixed(ScreenPosition{x, y}, this->cam);

for (auto &button : this->buttons) {
    button.render(this->cam);
}

this->window.render();
}

void GUI::CreateGameWindow::buttonPressed(GUI::ScreenPosition sp) {
    if (this->buttons[0].inside(sp)) {
        this->notify(*this, Event::LevelSelected);
    }

    if (this->buttons[1].inside(sp)) {
        this->buttonSelected = (this->buttonSelected + 1) % this->levelsInfo.size();
    }

    if (this->buttons[2].inside(sp)) {
        this->buttonSelected = (this->buttonSelected == 0) ? this->levelsInfo.size() - 1 : this->buttonSelected - 1;
    }
}

void GUI::CreateGameWindow::appendCharacter(char *text) {
}

void GUI::CreateGameWindow::handleKeyDown(SDL_Keycode key) {
}
```

jun 29, 18 16:28

CreateGameWindow.h

Page 1/1

```

//
// Created by rodrigo on 23/06/18.
//

#ifndef INC_4_WORMS_CREATEGAMEWINDOW_H
#define INC_4_WORMS_CREATEGAMEWINDOW_H

#include <vector>

#include "Window.h"
#include "Font.h"
#include "GameStateMsg.h"
#include "GameWindow.h"
#include "Button.h"

#define SELECT_LEVEL_MSG "Select"
#define LEVEL_MSG "Level"
#define PLAYERS_MSG "Players"
#define NEXT_LEVEL_MSG "Next"
#define PREVIOUS_LEVEL_MSG "Previous"

namespace GUI {
    class CreateGameWindow : public GameWindow {
    public:
        std::vector<IO::LevelInfo> &levelsInfo;

        explicit CreateGameWindow(GUI::Window &window, GUI::Font &font, GUI::Camera &cam,
                                   std::vector<IO::LevelInfo> &levelsInfo);

        void start() override;
        void render() override;
        void handleKeyDown(SDL_Keycode key) override;
        void appendCharacter(char text[32]) override;
        void buttonPressed(ScreenPosition sp) override;

    private:
        std::vector<Button> buttons;
        int levelInfoSize{30};
    };
}

#endif //INC_4_WORMS_CREATEGAMEWINDOW_H

```

jun 29, 18 16:28	Dead.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 28/05/18 */ #include <iostream> #include "Dead.h" Worm::Dead::Dead() : State(StateID::Dead) {} Worm::Dead::~Dead() {} void Worm::Dead::update(float dt) { } IO::PlayerInput Worm::Dead::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::cluster(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 29, 18 16:28	Dead.cpp	Page 2/2
<pre> } IO::PlayerInput Worm::Dead::mortar(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Dead::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

Dead.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#ifndef __Dead_H__
#define __Dead_H__

#include "WormState.h"

namespace Worm {
class Dead : public State {
public:
    Dead();
    ~Dead();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __Dead_H__

```

jun 26, 18 2:39	Die.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 28/05/18 */ #include "Die.h" Worm::Die::Die() : State(StateID::Die) {} Worm::Die::~~Die() {} void Worm::Die::update(float dt) {} IO::PlayerInput Worm::Die::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	Die.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Die::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

Die.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#ifndef __Die_H__
#define __Die_H__

#include "WormState.h"

namespace Worm {
class Die : public State {
public:
    Die();
    ~Die();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __Die_H__

```

jun 26, 18 2:39	Drowning.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 29/05/18 */ #include "Drowning.h" Worm::Drowning::Drowning() : State(StateID::Drowning) {} Worm::Drowning::~Drowning() {} void Worm::Drowning::update(float dt) {} IO::PlayerInput Worm::Drowning::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	Drowning.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Drowning::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

Drowning.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 29/05/18
 */

#ifndef __Drown_H__
#define __Drown_H__

#include "WormState.h"

namespace Worm {
class Drowning : public State {
public:
    Drowning();
    ~Drowning();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __Drown_H__

```

jun 26, 18 2:39

Dynamite.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#include "Dynamite.h"

Worm::Dynamite::Dynamite(const GUI::GameTextureManager &tex)
    : Weapon(tex, GUI::GameTextures::WormDynamite, DYNAMITE_CENTER_FRAME, Weapon
ID::WDynamite) {}

void Worm::Dynamite::update(float dt) {
    this->weaponAnimation.update(dt);
}

void Worm::Dynamite::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip
&flip) {
    this->weaponAnimation.render(p, cam, flip);
}

void Worm::Dynamite::setAngle(float angle, Worm::Direction d) {}

void Worm::Dynamite::startShot() {}

void Worm::Dynamite::endShot() {}

bool Worm::Dynamite::positionSelected() {
    return false;
}
```

jun 26, 18 2:39

Dynamite.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#ifndef __DYNAMITE_H__
#define __DYNAMITE_H__

#include "Weapon.h"

#define DYNAMITE_CENTER_FRAME 0

namespace Worm {
class Dynamite : public Weapon {
public:
    explicit Dynamite(const GUI::GameTextureManager &textureManager);
    ~Dynamite() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;
};
} // namespace Worm

#endif //__DYNAMITE_H__

```

jun 26, 18 7:40

Explosion.cpp

Page 1/1

```
//  
// Created by rodrigo on 2/06/18.  
//  
  
#include "Explosion.h"  
  
Worm::Explosion::Explosion(const GUI::GameTextureManager &texture_mgr) : texture_mgr(texture_mgr) {  
    this->animations.emplace_back(this->texture_mgr.get(GUI::GameTextures::Explosion));  
    this->animations.back().setAnimateOnce();  
}  
  
// TODO make observer in client side to clean exploded bullet  
void Worm::Explosion::update(float dt) {  
    for (auto &animation : this->animations) {  
        animation.update(dt);  
        this->explosionFinished = animation.finished();  
    }  
}  
  
void Worm::Explosion::render(GUI::Camera &cam) {  
    for (auto &animation : this->animations) {  
        animation.render(this->position, cam, SDL_FLIP_HORIZONTAL);  
    }  
}  
  
bool Worm::Explosion::finished() {  
    return this->explosionFinished;  
}
```

jun 26, 18 2:39

Explosion.h

Page 1/1

```
//  
// Created by rodrigo on 2/06/18.  
//  
  
#ifndef INC_4_WORMS_EXPLOSION_H  
#define INC_4_WORMS_EXPLOSION_H  
  
#include <Animation.h>  
#include <vector>  
#include "../GameTextures.h"  
  
namespace Worm {  
class Explosion {  
public:  
    explicit Explosion(const GUI::GameTextureManager &texture_mgr);  
    ~Explosion() = default;  
    void update(float dt);  
    void render(GUI::Camera &cam);  
    GUI::Position position{0, 0};  
    bool finished();  
  
private:  
    const GUI::GameTextureManager &texture_mgr;  
    std::vector<GUI::Animation> animations;  
    bool explosionFinished{false};  
};  
}  
  
#endif // INC_4_WORMS_EXPLOSION_H
```

jun 26, 18 2:39	Falling.cpp	Page 1/2
<pre>// // Created by rodrigo on 3/06/18. // #include "Falling.h" Worm::Falling::Falling() : State(StateID::Falling) {} Worm::Falling::~~Falling() {} void Worm::Falling::update(float dt) {} IO::PlayerInput Worm::Falling::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::mortar(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39	Falling.cpp	Page 2/2
<pre>} IO::PlayerInput Worm::Falling::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Falling::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39

Falling.h

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#ifndef INC_4_WORMS_FALLING_H
#define INC_4_WORMS_FALLING_H

#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class Falling : public State {
public:
    Falling();
    ~Falling();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // INC_4_WORMS_FALLING_H

```

jun 29, 18 16:28

GameBackgroundMusic.h

Page 1/1

```
//  
// Created by rodrigo on 25/06/18.  
//  
  
#ifndef INC_4_WORMS_GAMEBACKGROUNDMUSIC_H  
#define INC_4_WORMS_GAMEBACKGROUNDMUSIC_H  
  
#include "BackgroundMusicManager.h"  
#include "utils.h"  
  
namespace GUI {  
    /** Different kinds of background music. */  
    enum class GameBackgroundMusic {  
        Original,  
        MurderTrain  
    };  
  
    /** Specialized BackgroundMusicManager class. */  
    using GameBackgroundMusicManager = BackgroundMusicManager<GameBackgroundMusic,  
        Utils::EnumClassHash>;  
} // namespace GUI  
  
#endif //INC_4_WORMS_GAMEBACKGROUNDMUSIC_H
```


jun 29, 18 16:28

GameEndWindow.cpp

Page 1/1

```

//
// Created by rodrigo on 26/06/18.
//

#include "GameEndWindow.h"

GUI::GameEndWindow::GameEndWindow(GUI::Window &window, GUI::Font &font, GUI::Camera &cam, bool youWin) :
    GameWindow(window, font, cam) {
    this->gameEndResultMsg = youWin ? "You Win!" : "You Lose!";
}

void GUI::GameEndWindow::start() {
    while (!this->quit) {
        SDL_Event e;
        while (SDL_PollEvent(&e) != 0) {
            switch (e.type) {
                case SDL_QUIT: {
                    this->quit = true;
                    throw;
                    break;
                }
                default: {
                    break;
                }
            }
        }

        this->render();
    }
}

void GUI::GameEndWindow::render() {
    this->window.clear(SDL_Color{0xFF, 0xFF, 0xFF});

    SDL_Color black{0, 0, 0};

    Text gameResult{this->font};
    int x = this->window.getWidth() / 2;
    int y = this->window.getHeight() / 2;
    gameResult.set(this->gameEndResultMsg, black, 50);
    gameResult.renderFixed(ScreenPosition{x, y}, this->cam);

    this->window.render();
}

void GUI::GameEndWindow::buttonPressed(GUI::ScreenPosition sp) {
}

void GUI::GameEndWindow::appendCharacter(char *text) {
}

void GUI::GameEndWindow::handleKeyDown(SDL_Keycode key) {
}

```

jun 29, 18 16:28

GameEndWindow.h

Page 1/1

```

//
// Created by rodrigo on 26/06/18.
//

#ifndef INC_4_WORMS_GAMEENDWINDOW_H
#define INC_4_WORMS_GAMEENDWINDOW_H

#include <vector>

#include "Window.h"
#include "Font.h"
#include "GameStateMsg.h"
#include "GameWindow.h"
#include "Button.h"

namespace GUI {
    class GameEndWindow : public GameWindow {
    public:
        explicit GameEndWindow(GUI::Window &window, GUI::Font &font, GUI::Camera
        &cam, bool youWin);

        void start() override;
        void render() override;
        void handleKeyDown(SDL_Keycode key) override;
        void appendCharacter(char text[32]) override;
        void buttonPressed(ScreenPosition sp) override;

    private:
        std::vector<Button> buttons;
        int textSize{50};
        std::string gameEndResultMsg;
    };
}

#endif //INC_4_WORMS_GAMEENDWINDOW_H

```

jun 26, 18 2:39

GameSoundEffects.h

Page 1/1

```
//  
// Created by rodrigo on 4/06/18.  
//  
  
#ifndef INC_4_WORMS_GAMESOUNDEFFECTS_H  
#define INC_4_WORMS_GAMESOUNDEFFECTS_H  
  
#include "SoundEffectManager.h"  
#include "utils.h"  
  
namespace GUI {  
    /** Different kinds of sound effects. */  
    enum class GameSoundEffects {  
        WalkCompress,  
        Explosion,  
        WormLanding,  
        WormDrowning,  
        Splash,  
        WormJump,  
        WormBackFlip,  
        WormHit,  
        WormDie,  
        Holy,  
        AirStrike,  
        Teleport,  
        Shot,  
        Banana  
    };  
  
    /** Specialized SoundEffectManager class. */  
    using GameSoundEffectManager = SoundEffectManager<GameSoundEffects, Utils::EnumC  
lassHash>;  
} // namespace GUI  
  
#endif // INC_4_WORMS_GAMESOUNDEFFECTS_H
```

jun 26, 18 7:40	GameTextures.h	Page 1/2
<pre> #ifndef GAME_TEXTURES_H_ #define GAME_TEXTURES_H_ #include "TextureManager.h" #include "utils.h" namespace GUI { /** Different kinds of textures. */ enum class GameTextures { WormWalk, WormIdle, LongGirder, ShortGirder, StartJump, Jumping, EndJump, BackFlipping, Bazooka, Missile, Fly, Die, Dead, Sliding, StaticBackground, Background1, Background2, Background3, WormGrenade, Grenade, WormCluster, Cluster, Mortar, Bazooka2, WormBanana, Banana, WormHoly, Holy, Explosion, Flame, Smoke, Falling, Scope, PowerBar, Fragment, BazookaIcon, GrenadeIcon, ClusterIcon, MortarIcon, BananaIcon, HolyIcon, WormAirAttack, AirMissile, AirIcon, WormDynamite, Dynamite, DynamiteIcon, WormTeleport, WormTeleporting, TeleportIcon, WormBaseballBat, WormBaseballBatting, BaseballBatIcon, </pre>		

jun 26, 18 7:40	GameTextures.h	Page 2/2
<pre> WindLeft, WindRight, CurrentPlayerArrow, Water, }; /** Specialized TextureManager class. */ using GameTextureManager = TextureManager<GameTextures, Utils::EnumClassHash>; } // namespace GUI #endif </pre>		

jun 29, 18 16:28

GameWindow.cpp

Page 1/1

```
//  
// Created by rodrigo on 19/06/18.  
//  
  
#include <Font.h>  
#include <Camera.h>  
#include "GameWindow.h"  
  
GUI::GameWindow::GameWindow(Window &window, Font &font, Camera &cam) :  
    window(window),  
    font(font),  
    cam(cam) {  
  
}
```

jun 29, 18 16:28	GameWindow.h	Page 1/2
------------------	---------------------	----------

```

//
// Created by rodrigo on 19/06/18.
//

#ifndef INC_4_WORMS_GAMEWINDOW_H
#define INC_4_WORMS_GAMEWINDOW_H

#include <vector>

#include "Button.h"
#include "Camera.h"
#include "Font.h"
#include "Subject.h"
#include "Window.h"

#define ASSETS_PATH "/var/Worms/assets"

namespace GUI {
    struct TextField {
        TextField(std::string &text, ScreenPosition sp, int height, int width, Font &font) :
            inputText(sp, height, width, text, font),
            focus(false) {};

        void selected(ScreenPosition sp) {
            this->focus = inputText.inside(sp);
        };

        void render(GUI::Camera &cam) {
            this->inputText.render(cam);
        };

        void appendCharacter(char *text) {
            if (this->emptyString) {
                this->inputText.msg = text;
                this->emptyString = false;
            } else {
                this->inputText.msg += text;
            }
        };

        void backSpace() {
            if (!this->emptyString) {
                this->inputText.msg.pop_back();
                if (this->inputText.msg.length() == 0) {
                    this->inputText.msg = "";
                    this->emptyString = true;
                }
            }
        };

        Button inputText;
        bool focus;

    private:
        bool emptyString{true};
    };

    class GameWindow : public Subject {
    public:

```

jun 29, 18 16:28	GameWindow.h	Page 2/2
------------------	---------------------	----------

```

        uint8_t buttonSelected{0};

        explicit GameWindow(Window &window, Font &font, Camera &cam);

        virtual void start() = 0;
        virtual void render() = 0;
        virtual void handleKeyDown(SDL_Keycode key) = 0;
        virtual void appendCharacter(char text[32]) = 0;
        virtual void buttonPressed(ScreenPosition sp) = 0;

    protected:
        Window &window;
        Font &font;
        Camera &cam;
        std::vector<TextField> textFields;
        bool quit{false};
    };
}

#endif //INC_4_WORMS_GAMEWINDOW_H

```

jun 26, 18 2:39

Grenade.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include <cmath>

#include "Grenade.h"

Worm::Grenade::Grenade(const GUI::GameTextureManager &tex)
: Weapon(tex, GUI::GameTextures::WormGrenade, GRENADE_CENTER_FRAME, WeaponID
::WGrenade),
  scope(this->textureMgr),
  powerBar(this->textureMgr) {}

void Worm::Grenade::update(float dt) {
    this->weaponAnimation.update(dt);
    this->scope.update(dt);
    this->powerBar.update(dt);
}

void Worm::Grenade::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip
&flip) {
    this->weaponAnimation.render(p, cam, flip);
    this->scope.render(p, cam, flip);
    this->powerBar.render(p, cam, flip);
}

void Worm::Grenade::setAngle(float angle, Worm::Direction d) {
    this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->ce
nterFrame);
    this->scope.setAngle(angle, d);
    this->powerBar.setAngle(angle, d);
}

void Worm::Grenade::startShot() {
    this->powerBar.startShot();
}

void Worm::Grenade::endShot() {
    this->powerBar.endShot();
}

bool Worm::Grenade::positionSelected() {
    return false;
}

```

jun 26, 18 2:39

Grenade.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __GRENADE_H__
#define __GRENADE_H__

#include <Camera.h>

#include "../GameTextures.h"
#include "Direction.h"
#include "PowerBar.h"
#include "Scope.h"
#include "Weapon.h"

#define GRENADE_CENTER_FRAME 15

namespace Worm {
class Grenade : public Weapon {
public:
    explicit Grenade(const GUI::GameTextureManager &textureManager);
    ~Grenade() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
    ::Weapon::PowerBar powerBar;
};
} // namespace Worm

#endif //__GRENADE_H__

```


jun 29, 18 16:28 **GUIGame.cpp** Page 1/12

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 17/05/18.
 */

#include <SDL2/SDL.h>
#include <unistd.h>
#include <cmath>
#include <iostream>
#include <sstream>

#include "GameStateMsg.h"
#include "GameWindow.h"
#include "GUIGame.h"
#include "Stream.h"
#include "Text.h"
#include "Weapons/Bullet.h"
#include "Window.h"
#include "WrapTexture.h"

// TODO DEHARDCODE
GUI::Game::Game(Window &w, Worms::Stage &&stage, std::vector<std::string> &backg
roundPaths, ClientSocket &socket,
                std::uint8_t team)
    : window(w),
      texture_mgr(w.getRenderer()),
      sound_effect_mgr(),
      stage(stage),
      cam(w, this->scale, this->stage.getWidth(), this->stage.getHeight()),
      font(std::string(ASSETS_PATH) + "/fonts/gruen_lemonograf.ttf", 28),
      armory(this->texture_mgr, this->cam, this->font),
      socket(socket),
      team(team),
      wind(this->texture_mgr, this->cam),
      water(this->texture_mgr) {

    this->loadTextureManager();
    this->loadSoundManager();
    this->loadBackgroundManager();

    /* updates the armory */
    this->armory.loadWeapons();
    /* allocates space in the array to avoid the player addresses from changing */
    int num_worms = 0;
    this->worms.reserve(stage.getWorms().size());
    for (const auto &wormData : this->stage.getWorms()) {
        this->worms.emplace_back(num_worms, this->texture_mgr, this->sound_effec
t_mgr);
        this->snapshot.positions[num_worms * 2] = wormData.position.x;
        this->snapshot.positions[num_worms * 2 + 1] = wormData.position.y;
        this->snapshot.wormsHealth[num_worms] = wormData.health;
        num_worms += 1;
    }

    this->snapshot.num_worms = num_worms;
    // this->snapshot.processingInputs = true;

    this->teamColors.push_back(SDL_Color{0xFF, 0, 0});
    this->teamColors.push_back(SDL_Color{0, 0xFF, 0});
    this->teamColors.push_back(SDL_Color{0, 0, 0xFF});

```

jun 29, 18 16:28 **GUIGame.cpp** Page 2/12

```

    this->teamColors.push_back(SDL_Color{0xFF, 0, 0xFF});

    this->currentPlayerArrow = std::unique_ptr<GUI::Animation>(
        new GUI::Animation(this->texture_mgr.get(GUI::GameTextures::CurrentPlaye
rArrow), false));
    this->inputThread = std::thread([this] { this->inputWorker(); });
    this->outputThread = std::thread([this] { this->outputWorker(); });

    this->backgroundMusicPlayer =
        std::unique_ptr<GUI::BackgroundMusicPlayer>(new GUI::BackgroundMusic
Player{
            this->background_music_mgr.get(GUI::GameBackgroundMusic::Mur
derTrain));
    this->backgroundMusicPlayer->play();
}

GUI::Game::~Game() {
    this->exit();
    this->outputThread.join();
    this->inputThread.join();
}

void GUI::Game::inputWorker() {
    IO::GameStateMsg msg;
    try {
        while (!this->quit) {
            /* receives the size of the msg */
            std::uint32_t size(0);
            socket.receive((char *)&size, sizeof(std::uint32_t));
            size = ntohl(size);

            std::vector<char> buffer(size, 0);
            /* reads the raw data from the buffer */
            socket.receive(buffer.data(), size);

            std::string buff(buffer.data(), size);

            /* sets the struct data from the buffer */
            msg.deserialize(buff);
            this->snapshotBuffer.set(msg);
            this->snapshotBuffer.swap();
        }
    } catch (const std::exception &e) {
        std::cerr << "GUI::Game::inputWorker:" << e.what() << std::endl;
    } catch (...) {
        std::cerr << "Unknown error in GUI::Game::inputWorker()" << std::endl;
    }
}

void GUI::Game::outputWorker() {
    IO::PlayerMsg msg;
    try {
        while (!this->quit) {
            this->output.pop(msg, true);
            std::string buff = msg.serialize();
            std::uint32_t size = buff.size();
            std::uint32_t netSize = htonl(size);

            this->socket.send((char *)&netSize, sizeof(std::uint32_t));
            this->socket.send(buff.c_str(), size);
        }
    } catch (const std::exception &e) {

```

jun 29, 18 16:28

GUIGame.cpp

Page 3/12

```

        std::cerr << "GUI::Game::outputWorker:" << e.what() << std::endl;
    }
}

//void GUI::Game::inputWorker() {
//    IO::GameStateMsg msg;
//    char *buffer = new char[msg.getSerializedSize()];
//    try {
//        while (!this->quit) {
//            this->socket.receive(buffer, msg.getSerializedSize());
//            msg.deserialize(buffer, msg.getSerializedSize());
//            this->snapshotBuffer.set(msg);
//            this->snapshotBuffer.swap();
//        }
//    } catch (const std::exception &e) {
//        std::cerr << "GUI::Game::inputWorker:" << e.what() << std::endl;
//    } catch (...) {
//        std::cerr << "Unknown error in GUI::Game::inputWorker()" << std::endl;
//    }
//    delete[] buffer;
//}

//void GUI::Game::outputWorker() {
//    IO::PlayerMsg msg;
//    char *buffer = new char[msg.getSerializedSize()];
//    try {
//        while (!this->quit) {
//            this->output.pop(msg, true);
//            msg.serialize(buffer, msg.getSerializedSize());
//            this->socket.send(buffer, msg.getSerializedSize());
//        }
//    } catch (const std::exception &e) {
//        std::cerr << "GUI::Game::outputWorker:" << e.what() << std::endl;
//    } catch (...) {
//        std::cerr << "Unknown error in GUI::Game::outputWorker()" << std::endl;
//    }
//    delete[] buffer;
//}

void GUI::Game::start() {
    try {
        uint32_t prev = SDL_GetTicks();
        while (!this->quit) {
            /* updates the snapshot */
            this->snapshot = this->snapshotBuffer.get();
            if (!this->snapshot.gameEnded) {
                Worm::Worm &cur = this->worms[this->snapshot.currentWorm];

                /* handle events on queue */
                SDL_Event e;
                while (SDL_PollEvent(&e) != 0) {
                    switch (e.type) {
                        case SDL_QUIT:
                            this->exit();
                            break;
                        case SDL_KEYDOWN:
                            if (this->snapshot.processingInputs &&

```

jun 29, 18 16:28

GUIGame.cpp

Page 4/12

```

        this->team == this->snapshot.currentTeam) {
            cur.handleKeyDown(e.key.keysym.sym, &this->output);
        }
        break;
    case SDL_KEYUP:
        if (this->snapshot.processingInputs &&
            this->team == this->snapshot.currentTeam) {
            cur.handleKeyUp(e.key.keysym.sym, &this->output);
        }
        break;
    case SDL_MOUSEBUTTONDOWN: {
        if (this->snapshot.processingInputs &&
            this->team == this->snapshot.currentTeam) {
            int x, y;
            SDL_GetMouseState(&x, &y);
            GUI::Position global =
                this->cam.screenToGlobal(GUI::ScreenPosition
                    (x, y));

            cur.mouseButtonDown(global, &this->output);
            break;
        }
    }
    default:
        break;
}

/* synchronizes the worms states with the server's */
for (std::size_t i = 0; i < this->worms.size(); i++) {
    this->worms[i].setState(this->snapshot.stateIDs[i]);
    this->worms[i].setWeapon((i != this->snapshot.currentWorm)
        ? Worm::WeaponID::WNone
        : this->snapshot.activePlayerWeapon);
}

if (cur.getState() == Worm::StateID::Still &&
    cur.getWeaponID() != Worm::WeaponID::WNone) {
    cur.setWeaponAngle(this->snapshot.activePlayerAngle);
}

if (this->snapshot.bulletsQuantity == 0 && this->snapshot.player
UsedTool) {
    this->bullets.erase(this->bullets.begin(), this->bullets.end
());
    this->explodedQuantity = 0;
    this->worms[this->snapshot.currentWorm].reset();
}

if (this->snapshot.bulletsQuantity > 0) {
    for (int i = this->bullets.size(); i < this->snapshot.bullet
sQuantity; i++) {
        std::shared_ptr<Ammo::Bullet> p(
            new Ammo::Bullet(this->texture_mgr, this->sound_effe
ct_mgr,
                this->snapshot.bulletType[i]));
        this->bullets.emplace_back(p);
    }
    int i = 0;
    for (auto &bullet : this->bullets) {
        if (this->snapshot.bulletType[i] == Worm::WeaponID::WExp
lode &&

```

jun 29, 18 16:28

GUIGame.cpp

Page 5/12

```

        !bullet->exploding()) {
            bullet->madeImpact();
            this->explodedQuantity++;
        }
        bullet->setAngle(this->snapshot.bulletsAngle[i++]);
    }

    uint32_t current = SDL_GetTicks();
    float dt = static_cast<float>(current - prev) / 1000.0f;
    prev = current;

    this->handleCamera(dt);
    this->update(dt);

    this->render();
} else {
    this->youWin = this->snapshot.winner == this->team;
    this->quit = true;
}
}
} catch (std::exception &e) {
    std::cerr << e.what() << std::endl << "In GUI::Game::start" << std::endl;
} catch (...) {
    std::cerr << "Unkown error in GUI::Game::start()." << std::endl;
}

void GUI::Game::update(float dt) {
    for (auto &worm : this->worms) {
        worm.health = this->snapshot.wormsHealth[static_cast<int>(worm.id)];
        worm.direction = this->snapshot.wormsDirection[static_cast<int>(worm.id)];
        worm.update(dt);
    }
    if (this->snapshot.waitingForNextTurn) {
        this->armory.update(this->snapshot);
        this->currentPlayerArrow->update(dt);
    } else {
        this->currentPlayerArrow->setFrame(0);
    }

    this->cam.update(dt);

    for (auto &bullet : this->bullets) {
        bullet->update(dt);
    }

    this->water.update(dt);
}

void GUI::Game::render() {
    this->renderBackground();

    for (uint8_t i = 0; i < this->snapshot.num_worms; i++) {
        float cur_x = this->snapshot.positions[i * 2];
        float cur_y = this->snapshot.positions[i * 2 + 1];

        GUI::Position p{cur_x, cur_y};
        this->worms[i].setPosition(p);
        this->worms[i].render(p, this->cam);
    }
}

```

jun 29, 18 16:28

GUIGame.cpp

Page 6/12

```

    for (auto &girder : this->stage.getGirders()) {
        const GUI::Texture &texture = this->texture_mgr.get(GUI::GameTextures::LongGirder);

        GUI::WrapTexture wt{texture, girder.length, girder.height};
        wt.render(GUI::Position{girder.pos.x, girder.pos.y}, girder.angle, this->cam);
    }

    int i = 0, j = 0;
    for (auto &bullet : this->bullets) {
        float local_x = this->snapshot.bullets[i++];
        float local_y = this->snapshot.bullets[i++];
        if (!bullet->exploding()) {
            bullet->setAngle(this->snapshot.bulletsAngle[j++]);
            bullet->setPosition(GUI::Position{local_x, local_y});
        }

        if (!bullet->exploded()) {
            bullet->render(GUI::Position{local_x, local_y}, this->cam);
        }
    }

    /* health bars are renderer after the worms so they appear on top */
    for (uint8_t i = 0; i < this->snapshot.num_worms; i++) {
        float cur_x = this->snapshot.positions[i * 2];
        float cur_y = this->snapshot.positions[i * 2 + 1];
        if (this->worms[i].getState() != Worm::StateID::Dead) {
            Text health{this->font};
            health.setBackground(SDL_Color{0, 0, 0});
            health.set(std::to_string(static_cast<int>(this->worms[i].health)),
                this->teamColors[this->snapshot.wormsTeam[i]], 20);
            health.render(GUI::Position{cur_x, cur_y + 2.2f}, this->cam);
        }
    }

    this->water.render(this->cam);

    this->renderStatic();

    this->window.render();
}

/**
 * @brief interrupts all current game operations and leaves the main loop.
 */
void GUI::Game::exit() {
    this->quit = true;
    this->output.close();
    this->socket.shutdown();
}

/**
 * @brief Renders the background images using a parallax effect.
 */
void GUI::Game::renderBackground() {
    SDL_Color bgColor{this->stage.backgroundColor.r, this->stage.backgroundColor.g, this->stage.backgroundColor.b};
    this->window.clear(bgColor);
}

```

jun 29, 18 16:28

GUIGame.cpp

Page 7/12

```

/* draws moving image further in the background */
const Texture &Bg1Tex = this->texture_mgr.get(GameTextures::Background1);
// TODO: use the stage size
WrapTexture bg1(Bg1Tex, this->stage.getWidth(), Bg1Tex.getHeight() / this->cam.getScale());

Position pos{0.0f, (Bg1Tex.getHeight() / this->cam.getScale()) / 2};
pos.x += this->cam.getPosition().x * 0.8f;
bg1.render(pos, this->cam);

/* draws a moving image in the background at intermediate distance */
const Texture &Bg2Tex = this->texture_mgr.get(GameTextures::Background2);
// TODO: use the stage size
WrapTexture bg2(Bg2Tex, this->stage.getWidth(), Bg2Tex.getHeight() / this->cam.getScale());

pos = {0.0f, (Bg2Tex.getHeight() / this->cam.getScale()) / 2};
pos.x += this->cam.getPosition().x * 0.6f;
bg2.render(pos, this->cam);

/* draws a moving image in the background at a closer distance */
const Texture &Bg3Tex = this->texture_mgr.get(GameTextures::Background3);
// TODO: use the stage size
WrapTexture bg3(Bg3Tex, this->stage.getWidth(), Bg3Tex.getHeight() / this->cam.getScale());

pos = {0.0f, (Bg3Tex.getHeight() / this->cam.getScale()) / 2};
pos.x += this->cam.getPosition().x * 0.25f;
bg3.render(pos, this->cam);
}

/**
 * @brief Draws the game controls.
 */
void GUI::Game::renderStatic() {

    /* render the arrow to notify the current player when waiting for next turn */
    if (this->snapshot.waitingForNextTurn) {
        float cur_x = this->snapshot.positions[this->snapshot.currentWorm * 2];
        float cur_y = this->snapshot.positions[this->snapshot.currentWorm * 2 + 1];

        GUI::Position position = GUI::Position{cur_x, cur_y + 4.4f};
        this->currentPlayerArrow->render(position, this->cam, SDL_FLIP_NONE);
    }

    /* health bars of the team */
    uint8_t numTeams = this->snapshot.num_teams;
    int textHeight = 25;
    for (uint8_t i = 0; i < numTeams; i++) {
        Text health(this->font);
        std::ostringstream oss;
        oss << "Team " << i + 1 << ": " << this->snapshot.teamHealths[i];

        health.setBackground(SDL_Color{0, 0, 0});
        health.set(oss.str(), this->teamColors[i], textHeight);
        int x = this->window.getWidth() / 2;
        int y = this->window.getHeight() - (textHeight * (numTeams - i));
        health.renderFixed(ScreenPosition{x, y}, this->cam);
    }
}

```

jun 29, 18 16:28

GUIGame.cpp

Page 8/12

```

/* displays the remaining turn time */
std::int16_t turnTimeLeft =
    this->snapshot.currentPlayerTurnTime - this->snapshot.elapsedTurnSeconds;

turnTimeLeft = (turnTimeLeft < 0) ? 0 : turnTimeLeft;

int x = this->window.getWidth() / 2;
int y = 20;

SDL_Color color = {0, 0, 0};
Text text(this->font);
text.set(std::to_string(turnTimeLeft), color);
text.renderFixed(ScreenPosition{x, y}, this->cam);

/* renders armory */
this->armory.render();

this->wind.render(this->snapshot.windIntensity, this->window.getWidth());
}

/**
 * @brief Handles the camera actions.
 *
 * @param dt Seconds elapsed since the last call to this function.
 */
void GUI::Game::handleCamera(float dt) {
    this->lastCameraUpdate += dt;

    /* checks the mouse to see if the user wishes to move the camera */
    int mx, my;
    SDL_GetMouseState(&mx, &my);

    const float cameraSpeed = 15.0f;
    const int cameraMargin = 50;

    /* checks if the camera should be moved horizontally */
    if (this->window.containsMouse()) {
        if (mx < cameraMargin) {
            auto p = this->cam.getPosition() - GUI::Position{cameraSpeed, 0.0f}
            * dt;
            this->cam.moveTo(this->cam.getPosition() - GUI::Position{cameraSpeed, 0.0f} * dt);
            this->lastCameraUpdate = 0.0f;
        } else if (mx > this->window.getWidth() - cameraMargin) {
            this->cam.moveTo(this->cam.getPosition() + GUI::Position{cameraSpeed, 0.0f} * dt);
            this->lastCameraUpdate = 0.0f;
        }

        /* checks if the camera should be moved vertically */
        if (my < cameraMargin) {
            this->cam.moveTo(this->cam.getPosition() + GUI::Position{0.0f, cameraSpeed} * dt);
            this->lastCameraUpdate = 0.0f;
        } else if (my > this->window.getHeight() - cameraMargin) {
            this->cam.moveTo(this->cam.getPosition() - GUI::Position{0.0f, cameraSpeed} * dt);
            this->lastCameraUpdate = 0.0f;
        }
    }

    /* if the user hasn't changed the camera in a while, it becomes automatic ag

```

jun 29, 18 16:28	GUIGame.cpp	Page 9/12
<pre> ain */ if (this->lastCameraUpdate < 2.0f) { return; } else { /* avoids overflow */ this->lastCameraUpdate = 2.0f; } /* move the camera to the current player */ if (this->snapshot.bulletsQuantity > this->explodedQuantity) { float cur_x(0); float cur_y(0); int i(0); for (int j = 0; i < this->snapshot.bulletsQuantity; i++) { if (this->snapshot.bulletType[i] != Worm::WExplode) { cur_x = this->snapshot.bullets[j++]; cur_y = this->snapshot.bullets[j]; break; } j += 2; } this->cam.moveTo(GUI::Position{cur_x, cur_y}); } else { float cur_follow_x = this->snapshot.positions[this->snapshot.currentWorm ToFollow * 2]; float cur_follow_y = this->snapshot.positions[this->snapshot.currentWorm ToFollow * 2 + 1]; /* move the camera to the current player */ this->cam.moveTo(GUI::Position{cur_follow_x, cur_follow_y}); } } void GUI::Game::loadTextureManager(){ std::string path(ASSETS_PATH); /* loads the required textures */ this->texture_mgr.load(GUI::GameTextures::CurrentPlayerArrow, path + "/img/Mi sc/arrowdnb.png", GUI::Color{0x40, 0x40, 0x80}); this->texture_mgr.load(GUI::GameTextures::WindLeft, path + "/img/Misc/windl.png" , GUI::Color{0x00, 0x00, 0x00}); this->texture_mgr.load(GUI::GameTextures::WindRight, path + "/img/Misc/windr.png" ", GUI::Color{0x00, 0x00, 0x00}); this->texture_mgr.load(GUI::GameTextures::WormWalk, path + "/img/Worms/wwalk2.p ng", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::WormIdle, path + "/img/Worms/wbrth1.pn g", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::LongGirder, path + "/img/Weapons/grdl4 .png", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::StartJump, path + "/img/Worms/wjump.p ng", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Jumping, path + "/img/Worms/wflyup.png" ", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::EndJump, path + "/img/Worms/wland2.png" </pre>		

jun 29, 18 16:28	GUIGame.cpp	Page 10/12
<pre> ", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::BackFlipping, path + "/img/Worms/wba ckflp.png", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Falling, path + "/img/Worms/wfall.png" , GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Bazooka, path + "/img/Worms/wbaz.png" , GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Fly, path + "/img/Worms/wfly1.png", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Die, path + "/img/Worms/wdie.png", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Sliding, path + "/img/Worms/wslided.png" ", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Dead, path + "/img/Misc/grave4.png", GUI::Color{0xc0, 0xc0, 0x80}); this->texture_mgr.load(GUI::GameTextures::Missile, path + "/img/Weapons/missile.pn g", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Explosion, path + "/img/Effects/circle25. png", GUI::Color{0x80, 0x80, 0xc0}); this->texture_mgr.load(GUI::GameTextures::Flame, path + "/img/Effects/flame1.png", GUI::Color{0x80, 0x80, 0xc0}); this->texture_mgr.load(GUI::GameTextures::Smoke, path + "/img/Effects/smkdrk20.png" ", GUI::Color{0xc0, 0xc0, 0x80}); this->texture_mgr.load(GUI::GameTextures::Background1, path + "/img/background/b g1.png", GUI::Color{0xff, 0xff, 0xff}); this->texture_mgr.load(GUI::GameTextures::Background2, path + "/img/background/b g2.png", GUI::Color{0xff, 0xff, 0xff}); this->texture_mgr.load(GUI::GameTextures::Background3, path + "/img/background/b g3.png", GUI::Color{0xff, 0xff, 0xff}); this->texture_mgr.load(GUI::GameTextures::WormGrenade, path + "/img/Worms/wthrg rn.png", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Grenade, path + "/img/Weapons/grenade.p ng", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::WormCluster, path + "/img/Worms/wthrc ls.png", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Cluster, path + "/img/Weapons/cluster.pn g", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::Mortar, path + "/img/Weapons/mortar.png" ", GUI::Color{0xc0, 0xc0, 0x80}); this->texture_mgr.load(GUI::GameTextures::Bazooka2, path + "/img/Worms/wbaz2.pn g", GUI::Color{0xc0, 0xc0, 0x80}); this->texture_mgr.load(GUI::GameTextures::Banana, path + "/img/Weapons/banana.png" ", GUI::Color{0x7f, 0x7f, 0xbb}); this->texture_mgr.load(GUI::GameTextures::WormBanana, path + "/img/Worms/wthrba </pre>		

jun 29, 18 16:28	GUIGame.cpp	Page 11/12
n.png",	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::Holy, path + "/img/Weapons/hgrenade.png",	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::WormHoly, path + "/img/Worms/wthrhgrd.png",	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::Scope, path + "/img/Misc/crshairb.png",	GUI::Color{0x40, 0x40, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::Scope, path + "/img/Misc/crshairb.png",	GUI::Color{0x40, 0x40, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::PowerBar, path + "/img/Effects/blob.png"	GUI::Color{0x80, 0x80, 0xc0});	
this ->texture_mgr.load(GUI::GameTextures::Fragment, path + "/img/Weapons/clustlet.png",	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::WormAirAttack, path + "/img/Worms/wa	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::AirMissile, path + "/img/Weapons/airmi	GUI::Color{0xc0, 0xc0, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::WormDynamite, path + "/img/Worms/wdy	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::Dynamite, path + "/img/Weapons/dynamit	GUI::Color{0x7f, 0x7f, 0xbb});	
this ->texture_mgr.load(GUI::GameTextures::WormBaseballBat, path + "/img/Worms/	GUI::Color{0xc0, 0xc0, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::WormBaseballBatting, path + "/img/	GUI::Color{0xc0, 0xc0, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::WormTeleport, path + "/img/Worms/wtel	GUI::Color{0xc0, 0xc0, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::WormTeleporting, path + "/img/Worms/	GUI::Color{0xc0, 0xc0, 0x80});	
this ->texture_mgr.load(GUI::GameTextures::BazookaIcon, path + "/img/Weapon Icon	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::GrenadeIcon, path + "/img/Weapon Icon	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::ClusterIcon, path + "/img/Weapon Icon	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::MortarIcon, path + "/img/Weapon Icons/	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::BananaIcon, path + "/img/Weapon Icons/	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::HolyIcon, path + "/img/Weapon Icons/hgr	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::AirIcon, path + "/img/Weapon Icons/airstr		

jun 29, 18 16:28	GUIGame.cpp	Page 12/12
this ->texture_mgr.load(GUI::GameTextures::DynamiteIcon,	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::BaseballBatIcon,	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::TeleportIcon,	GUI::Color{0x00, 0x00, 0x00});	
this ->texture_mgr.load(GUI::GameTextures::Water,	GUI::Color{0x00, 0x00, 0x00});	
void GUI::Game::loadSoundManager() {		
std::string path(ASSETS_PATH);		
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WalkCompress,	path + "/sound/Effects/Walk-Compress.wav");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WormJump,	path + "/sound/Soundbanks/JUMP1.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WormBackFlip,	path + "/sound/Soundbanks/JUMP2.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WormLanding,	path + "/sound/Effects/WormLanding.wav");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WormHit, path + "/sound/Sou	ndbanks/OUCH.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WormDrowning,	path + "/sound/Effects/UnderWaterLoop.wav");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::WormDie,	path + "/sound/Soundbanks/BYE.BYE.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::Splash, path + "/sound/Effect	s/Splash.wav");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::Explosion,	path + "/sound/Effects/Explosion1.wav");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::Holy,	path + "/sound/Effects/HOLYGRENAD.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::AirStrike,	path + "/sound/Effects/Airstrike.wav");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::Teleport,	path + "/sound/Effects/TELEPORT.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::Shot,	path + "/sound/Effects/ROCKETRELEASE.WAV");	
this ->sound_effect_mgr.load(GUI::GameSoundEffects::Banana,	path + "/sound/Effects/BananaImpact.wav");	
}		
void GUI::Game::loadBackgroundManager() {		
std::string path(ASSETS_PATH);		
this ->background_music_mgr.load(GUI::GameBackgroundMusic::Original,	path + "/sound/Background/background.wav");	
this ->background_music_mgr.load(GUI::GameBackgroundMusic::MurderTrain, path	+ "/sound/Background/MurderTrain.wav");	
}		

jun 29, 18 16:28	GUIGame.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #ifndef __GUIGame_H__ #define __GUIGame_H__ #include <atomic> #include <list> #include <thread> #include <vector> #include "Animation.h" #include "Armory.h" #include "Camera.h" #include "ClientSocket.h" #include "DoubleBuffer.h" #include "Font.h" #include "GameSoundEffects.h" #include "GameStateMsg.h" #include "GameTextures.h" #include "Stage.h" #include "Stream.h" #include "TextureManager.h" #include "Water.h" #include "Weapons/Bullet.h" #include "Weapons/Explosion.h" #include "Wind.h" #include "Window.h" #include "Worm.h" #include "BackgroundMusic.h" #include "GameBackgroundMusic.h" #include "BackgroundMusicPlayer.h" namespace GUI { using GameOutput = IO::Stream<IO::PlayerMsg>; class Game { public: bool youWin{false}; Game(Window &w, Worms::Stage &&stage, std::vector<std::string> &backgroundPa ths, ClientSocket &socket, std::uint8_t team); ~Game(); void start(); void update(float dt); void render(); void exit(); private: void renderStatic(); void renderBackground(); void handleCamera(float dt); void inputWorker(); void outputWorker(); std::atomic<bool> quit{false}; float scale{13.0f}; // pixels per meter float lastCameraUpdate{0.0f}; // pixels per meter </pre>		

jun 29, 18 16:28	GUIGame.h	Page 2/2
<pre> Window &window; GameTextureManager texture_mgr; GameSoundEffectManager sound_effect_mgr; GameBackgroundMusicManager background_music_mgr; std::vector<Worm::Worm> worms; Worms::Stage stage; std::list<std::shared_ptr<Ammo::Bullet>> bullets; Camera cam; Font font; SDL_Color backgroundColor{0xba, 0x8d, 0xc6}; std::vector<SDL_Color> teamColors; Armory armory; std::thread inputThread; std::thread outputThread; IO::DoubleBuffer<IO::GameStateMsg> snapshotBuffer; IO::GameStateMsg snapshot; GameOutput output; CommunicationSocket &socket; std::uint8_t team{0}; uint8_t explodedQuantity{0}; GUI::Wind wind; GUI::Water water; std::unique_ptr<Animation> currentPlayerArrow{nullptr}; std::unique_ptr<GUI::BackgroundMusicPlayer> backGroundMusicPlayer{nullptr}; void loadTextureManager(); void loadBackgroundManager(); void loadSoundManager(); }; } // namespace GUI #endif // __GUIGame_H__ </pre>		

jun 26, 18 2:39	Hit.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 28/05/18 */ #include "Hit.h" Worm::Hit::Hit() : State(StateID::Hit) {} Worm::Hit::~~Hit() {} void Worm::Hit::update(float dt) {} IO::PlayerInput Worm::Hit::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	Hit.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Hit::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

Hit.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#ifndef __Hit_H__
#define __Hit_H__

#include "WormState.h"

namespace Worm {
class Hit : public State {
public:
    explicit Hit();
    virtual ~Hit();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __Hit_H__

```

jun 26, 18 2:39

Holy.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include <cmath>

#include "Holy.h"

Worm::Holy::Holy(const GUI::GameTextureManager &tex)
: Weapon(tex, GUI::GameTextures::WormHoly, HOLY_CENTER_FRAME, WeaponID::WHoly),
  scope(this->textureMgr),
  powerBar(this->textureMgr) {}

void Worm::Holy::update(float dt) {
    this->weaponAnimation.update(dt);
    this->scope.update(dt);
    this->powerBar.update(dt);
}

void Worm::Holy::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) {
    this->weaponAnimation.render(p, cam, flip);
    this->scope.render(p, cam, flip);
    this->powerBar.render(p, cam, flip);
}

void Worm::Holy::setAngle(float angle, Worm::Direction d) {
    this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->centerFrame);
    this->scope.setAngle(angle, d);
    this->powerBar.setAngle(angle, d);
}

void Worm::Holy::startShot() {
    this->powerBar.startShot();
}

void Worm::Holy::endShot() {
    this->powerBar.endShot();
}

bool Worm::Holy::positionSelected() {
    return false;
}

```

jun 26, 18 2:39

Holy.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __HOLY_H__
#define __HOLY_H__

#include <vector>

#include "PowerBar.h"
#include "Scope.h"
#include "Weapon.h"

#define HOLY_CENTER_FRAME 15

namespace Worm {
class Holy : public Weapon {
public:
    explicit Holy(const GUI::GameTextureManager &textureManager);
    ~Holy() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
    ::Weapon::PowerBar powerBar;
};
} // namespace Worm

#endif //__HOLY_H__

```

jun 29, 18 16:28	JoinGameWindow.cpp	Page 1/2
------------------	---------------------------	----------

```

#include "JoinGameWindow.h"

const SDL_Color WHITE = {0xff, 0xff, 0xff};
const SDL_Color BLACK = {0, 0, 0};

const int TEXT_SIZE = 30;

GUI::JoinGameWindow::JoinGameWindow(Window &window, Font &font, Camera &cam,
std::vector<IO::GameInfo> &info)
: GameWindow(window, font, cam),
info(info),
gameName(font),
numPlayers(font),
prev("Previous", font),
next("Next", font),
join("Join", font) {
    int height = TEXT_SIZE * 3 / 2;
    this->prev.textColor = WHITE;
    this->prev.textSize = TEXT_SIZE;
    this->prev.position = {this->window.getWidth() / 4, this->window.getHeight() / 2};
    this->prev.height = height;
    this->prev.width = this->prev.msg.size() * 9 + 20;

    this->next.textColor = WHITE;
    this->next.textSize = TEXT_SIZE;
    this->next.position = {this->window.getWidth() * 3 / 4, this->window.getHeight() / 2};
    this->next.height = height;
    this->next.width = this->next.msg.size() * 9 + 20;

    this->join.textColor = WHITE;
    this->join.textSize = TEXT_SIZE;
    this->join.position = {this->window.getWidth() / 2, this->window.getHeight() * 3 / 4};
    this->join.height = height;
    this->join.width = this->join.msg.size() * 9 + 20;
}

/**
 * @brief Called when the window is started.
 */
void GUI::JoinGameWindow::start() {}

/**
 * @brief Renders the window.
 */
void GUI::JoinGameWindow::render() {
    this->window.clear(SDL_Color{0xFF, 0xFF, 0xFF});

    const ScreenPosition center{this->window.getWidth() / 2, this->window.getHeight() / 2};

    this->prev.render(this->cam);
    this->next.render(this->cam);

    if (this->info.size() > 0) {
        const IO::GameInfo &info = this->info.at(this->currentGameIndex);

        this->gameName.set("Game #" + std::to_string(info.gameID), BLACK, TEXT_S

```

jun 29, 18 16:28	JoinGameWindow.cpp	Page 2/2
------------------	---------------------------	----------

```

IZE * 2);
    this->gameName.renderFixed(center - ScreenPosition{0, this->window.getHeight() / 4},
                                this->cam);

    std::string msg =
        std::to_string(info.numCurrentPlayers) + "/" + std::to_string(info.numTotalPlayers);
    this->numPlayers.set(msg, BLACK, TEXT_SIZE * 2);
    this->numPlayers.renderFixed(center, this->cam);

    if (info.numCurrentPlayers < info.numTotalPlayers) {
        this->join.render(this->cam);
    }

    this->window.render();
}

/**
 * @brief Checks if a button was pressed.
 *
 * @param sp Position where there was a click.
 */
void GUI::JoinGameWindow::buttonPressed(ScreenPosition sp) {
    if (this->prev.inside(sp)) {
        if (this->currentGameIndex == 0) {
            this->currentGameIndex = static_cast<uint8_t>(this->info.size() - 1);
        } else {
            this->currentGameIndex--;
        }
    } else if (this->next.inside(sp)) {
        this->currentGameIndex = (this->currentGameIndex + 1) % this->info.size();
    } else if (this->join.inside(sp)) {
        const IO::GameInfo &info = this->info.at(this->currentGameIndex);
        if (info.numCurrentPlayers < info.numTotalPlayers) {
            this->notify(*this, Event::LobbyToJoinSelected);
        }
    }
}

/**
 * @brief Handles key press events.
 *
 * @param key Key pressed.
 */
void GUI::JoinGameWindow::handleKeyDown(SDL_Keycode key) {}

void GUI::JoinGameWindow::appendCharacter(char text[32]) {}

```

jun 29, 18 16:28

JoinGameWindow.h

Page 1/1

```

#ifndef JOIN_GAME_WINDOW_H_
#define JOIN_GAME_WINDOW_H_

#include <vector>
#include "../Button.h"
#include "../GameWindow.h"
#include "GameStateMsg.h"
#include "Text.h"
#include "Texture.h"

namespace GUI {
class JoinGameWindow : public GameWindow {
    public:
        JoinGameWindow(Window &window, Font &font, Camera &cam, std::vector<IO::Game
Info> &info);
        std::vector<IO::GameInfo> &info;
        uint8_t currentGameIndex{0};

        void start() override;
        void render() override;
        void handleKeyDown(SDL_Keycode key) override;
        void appendCharacter(char text[32]) override;
        void buttonPressed(ScreenPosition sp) override;

    private:
        Text gameName;
        Text numPlayers;
        Button prev;
        Button next;
        Button join;
};
} // namespace GUI

#endif

```

jun 26, 18 2:39	Land.cpp	Page 1/2
<pre>// // Created by rodrigo on 3/06/18. // #include "Land.h" Worm::Land::Land() : State(StateID::Land) {} Worm::Land::~~Land() {} void Worm::Land::update(float dt) {} IO::PlayerInput Worm::Land::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::mortar(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39	Land.cpp	Page 2/2
<pre>} IO::PlayerInput Worm::Land::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Land::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39

Land.h

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#ifndef INC_4_WORMS_LAND_H
#define INC_4_WORMS_LAND_H

#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class Land : public State {
public:
    Land();
    ~Land();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;
    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // INC_4_WORMS_LAND_H

```

jun 29, 18 16:28

LobbyAssistant.cpp

Page 1/4

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 17/06/18
 */

#include <iostream>
#include <GameStateMsg.h>
#include <SDL2/SDL.h>
#include <zconf.h>

#include "GameWindow.h"
#include "LobbyAssistant.h"
#include "Text.h"
#include "Window.h"
#include "SelectActionWindow.h"
#include "CreateGameWindow.h"
#include "WaitingPlayersWindow.h"
#include "Lobby/JoinGameWindow.h"
#include "ConnectionWindow.h"

GUI::LobbyAssistant::LobbyAssistant(Window &window) :
    window(window),
    font(std::string(ASSETS_PATH) + "/fonts/gruen_lemonograf.ttf", 28),
    cam(window, this->scale, 600, 600) {
    this->gameWindow = std::shared_ptr<GameWindow>(new ConnectionWindow{this->window, this->font, this->cam});
    this->gameWindow->addObserver(this);
}

void GUI::LobbyAssistant::run() {
    while (!this->quit) {
        SDL_Event e;
        while (SDL_PollEvent(&e) != 0) {
            switch (e.type) {
                case SDL_QUIT: {
                    this->quit = true;
                    this->exit = true;
                    break;
                }
                case SDL_KEYDOWN: {
                    this->gameWindow->handleKeyDown(e.key.keysym.sym);
                    break;
                }
                case SDL_KEYUP: {
                    break;
                }
                case SDL_TEXTINPUT: {
                    if(!((e.text.text[0] == 'c' || e.text.text[0] == 'C') && (e.text.text[0] == 'v' || e.text.text[0] == 'V') && SDL_GetModState() & KMOD_CTRL)) {
                        //Append character
                        this->gameWindow->appendCharacter(e.text.text);
                    }
                    break;
                }
                case SDL_MOUSEBUTTONDOWN: {
                    int x, y;
                    SDL_GetMouseState(&x, &y);
                    GUI::Position global = this->cam.screenToGlobal(GUI::ScreenPosition{x, y});
                    this->gameWindow->buttonPressed(ScreenPosition{x, y});
                }
            }
        }
    }
}

```

jun 29, 18 16:28

LobbyAssistant.cpp

Page 2/4

```

    }

    IO::ServerResponse sr{};
    if (this->serverStream.pop(sr, false)) {
        this->handleServerResponse(sr);
    }

    if (this->nextGameWindow) {
        this->gameWindow = this->nextGameWindow;
        this->nextGameWindow = nullptr;
    }

    if (this->gameWindow != nullptr) {
        this->gameWindow->render();
    }

    usleep(50 * 1000);
}

void GUI::LobbyAssistant::onNotify(Subject &subject, Event event) {
    switch (event) {
        case Event::ConnectionToServer: {
            auto connectionWindow = dynamic_cast<ConnectionWindow *>(this->gameWindow.get());
            ConnectionInfo info = connectionWindow->getConnectionInfo();
            ClientSocket socket(info.ip, info.port);
            this->communicationProtocol = std::shared_ptr<IO::CommunicationProtocol>(new IO::CommunicationProtocol(socket, &this->output, &this->serverStream));
            this->communicationProtocol->start();

            this->nextGameWindow = std::shared_ptr<GameWindow>(new SelectActionWindow{this->window, this->font, this->cam});
            this->nextGameWindow->addObserver(this);
            break;
        }
        case Event::CreateGame: {
            this->output << IO::ClientGUIMsg{IO::ClientGUIInput::startCreateGame};
            break;
        }
        case Event::LevelSelected: {
            auto createGameWindow = dynamic_cast<CreateGameWindow *>(this->gameWindow.get());
            this->communicationProtocol->levelToCreate = createGameWindow->buttonSelected;
            this->output << IO::ClientGUIMsg{IO::ClientGUIInput::levelSelected};
            this->nextGameWindow = std::shared_ptr<GameWindow>(new WaitingPlayerWindow{this->window,
                this->font,
                this->cam,
                createGameWindow->levelsInfo[createGameWindow->buttonSelected].playersQuantity});
            this->nextGameWindow->addObserver(this);
        }
    }
}

```


jun 29, 18 16:28

LobbyAssistant.cpp

Page 3/4

```

        break;
    }
    case Event::JoinGame: {
        this->output << IO::ClientGUIMsg{IO::ClientGUIInput::startJoinGame};
        break;
    }
    case Event::LobbyToJoinSelected: {
        auto joinGameWindow = dynamic_cast<JoinGameWindow *>(this->gameWindow.get());
        this->communicationProtocol->gameToJoin = joinGameWindow->currentGameIndex;
        this->communicationProtocol->levelOfGameToJoin = joinGameWindow->info[joinGameWindow->currentGameIndex].levelID;
        this->output << IO::ClientGUIMsg{IO::ClientGUIInput::joinGame};
        this->nextGameWindow = std::shared_ptr<GameWindow>(new WaitingPlayerWindow{this->window,
            this->font,
            this->cam,
            joinGameWindow->info[joinGameWindow->currentGameIndex].numTotalPlayers,
            joinGameWindow->info[joinGameWindow->currentGameIndex].numCurrentPlayers
        });
        this->nextGameWindow->addObserver(this);
        break;
    }
    default: {
        break;
    }
}

ClientSocket GUI::LobbyAssistant::getSocket() {
    return std::move(this->communicationProtocol->getSocket());
}

void GUI::LobbyAssistant::handleServerResponse(IO::ServerResponse &response) {
    switch (response.action) {
        case IO::ServerResponseAction::startGame: {
            this->levelPath = std::move(this->communicationProtocol->levelPath);
            this->backgroundPath = std::move(this->communicationProtocol->backgroundPath);
            this->output << IO::ClientGUIMsg{IO::ClientGUIInput::quit};
            this->quit = true;
            break;
        }
        case IO::ServerResponseAction::levelsInfo: {
            this->nextGameWindow = std::shared_ptr<GameWindow>(new CreateGameWindow{this->window,
                this->font,
                this->cam,
                this->communicationProtocol->levelsInfo});
            this->nextGameWindow->addObserver(this);
            break;
        }
        case IO::ServerResponseAction::gamesInfo: {
            this->nextGameWindow = std::shared_ptr<GameWindow>(new JoinGameWindow{this->window,
                this->font,
                this->cam,
                joinGameWindow->info[joinGameWindow->currentGameIndex].numTotalPlayers,
                joinGameWindow->info[joinGameWindow->currentGameIndex].numCurrentPlayers
            });
            this->nextGameWindow->addObserver(this);
            break;
        }
    }
}

```

jun 29, 18 16:28

LobbyAssistant.cpp

Page 4/4

```

w{this->window,
    this->font,
    this->cam,
    this->communicationProtocol->gamesInfo});
    this->nextGameWindow->addObserver(this);
    break;
}
    case IO::ServerResponseAction::playerConnected: {
        dynamic_cast<WaitingPlayersWindow *>(this->gameWindow.get())->playerConnected++;
        break;
    }
    case IO::ServerResponseAction::serverClosed: {
        this->quit = true;
        this->exit = true;
        this->gameWindow = nullptr;
        break;
    }
    default: {
        break;
    }
}

GUI::Font & GUI::LobbyAssistant::getFont() {
    return this->font;
}

GUI::Camera & GUI::LobbyAssistant::getCam() {
    return this->cam;
}

GUI::LobbyAssistant::~LobbyAssistant() {
    this->output.close();
    this->serverStream.close();
    if (this->communicationProtocol != nullptr) {
        this->communicationProtocol->stop();
        this->communicationProtocol->join();
    }
}

```

jun 29, 18 16:28

LobbyAssistant.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 17/06/18
 */

#ifndef __LOBBY_ASSISTANT_H__
#define __LOBBY_ASSISTANT_H__

#include <Protocol.h>
#include <memory>
#include <Stream.h>
#include <Font.h>
#include <Camera.h>
#include "ClientSocket.h"
#include "CommunicationProtocol.h"
#include "Observer.h"
#include "Thread.h"
#include "GameWindow.h"
#include "GameStateMsg.h"

namespace GUI { // HabÃ-a una forward declaration con GameWindow pero no hace falta parece.
    class LobbyAssistant : public Observer {
    public:
        std::string levelPath;
        std::vector<std::string> backgroundPath;
        bool exit{false};

        explicit LobbyAssistant(Window &window);
        ~LobbyAssistant();
        //TODO override
        void run();
        void onNotify(Subject &subject, Event event) override;

        ClientSocket getSocket();

        Font & getFont();

        Camera & getCam();

    private:
        Window &window;
        float scale{13.0f};
        bool quit{false};
        std::shared_ptr<GameWindow> gameWindow{nullptr};
        std::shared_ptr<GameWindow> nextGameWindow{nullptr};
        Font font;
        Camera cam;
        std::shared_ptr<IO::CommunicationProtocol> communicationProtocol;
        IO::Stream<IO::ClientGUIMsg> output;
        IO::Stream<IO::ServerResponse> serverStream;

        void handleServerResponse(IO::ServerResponse &response);
    };
} //namespace Worm

#endif //__LOBBY_ASSISTANT_H__

```

jun 29, 18 16:28

main.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */
#include <cstdlib>
#include <iostream>
#include <string>

#include "ClientSocket.h"
#include "GUIGame.h"
#include "LobbyAssistant.h"
#include "GameEndWindow.h"

int main(int argc, const char *argv[]) {
    if (argc != 1) {
        std::cout << "Usage: ./client" << std::endl;
        return EXIT_FAILURE;
    }

    try {
        GUI::Window window{};
        window.clear();
        GUI::LobbyAssistant lobby(window);
        lobby.run();

        if (!lobby.exit) {
            ClientSocket socket = std::move(lobby.getSocket());

            char buffer[1];
            socket.receive(buffer, sizeof(buffer));

            GUI::Game game(window, Worms::Stage::fromFile(lobby.levelPath), lobby.backgroundPath, socket,
                (std::uint8_t) buffer[0]);
            game.start();

            GUI::GameEndWindow gameEndWindow(window, lobby.getFont(), lobby.getCam(), game.youWin);
            gameEndWindow.start();
        }
    } catch (std::exception &e) {
        std::cerr << "In main()" << std::endl;
        std::cerr << e.what() << std::endl;
        return 1;
    } catch (...) {
        std::cerr << "Unkown error in main thread" << std::endl;
        return 1;
    }

    return 0;
}

```

jun 26, 18 2:39

Mortar.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include <cmath>

#include "Mortar.h"

Worm::Mortar::Mortar(const GUI::GameTextureManager &tex)
: Weapon(tex, GUI::GameTextures::Bazooka2, MORTAR_CENTER_FRAME, WeaponID::WM
ortar),
  scope(this->textureMgr),
  powerBar(this->textureMgr) {}

void Worm::Mortar::update(float dt) {
  this->weaponAnimation.update(dt);
  this->scope.update(dt);
  this->powerBar.update(dt);
}

void Worm::Mortar::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &
flip) {
  this->weaponAnimation.render(p, cam, flip);
  this->scope.render(p, cam, flip);
  this->powerBar.render(p, cam, flip);
}

void Worm::Mortar::setAngle(float angle, Worm::Direction d) {
  this->weaponAnimation.setFrame((int)std::ceil(angle / ANGLE_STEP) + this->ce
nterFrame);
  this->scope.setAngle(angle, d);
  this->powerBar.setAngle(angle, d);
}

void Worm::Mortar::startShot() {
  this->powerBar.startShot();
}

void Worm::Mortar::endShot() {
  this->powerBar.endShot();
}

bool Worm::Mortar::positionSelected() {
  return false;
}

```

jun 26, 18 2:39

Mortar.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __MORTAR_H__
#define __MORTAR_H__

#include <vector>

#include "PowerBar.h"
#include "Scope.h"
#include "Weapon.h"

#define MORTAR_CENTER_FRAME 16

namespace Worm {
class Mortar : public Weapon {
public:
    explicit Mortar(const GUI::GameTextureManager &textureManager);
    ~Mortar() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    ::Weapon::Scope scope;
    ::Weapon::PowerBar powerBar;
};
} // namespace Worm

#endif //__MORTAR_H__

```

jun 26, 18 2:39

PowerBar.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include "PowerBar.h"
#include "Weapon.h"

Weapon::PowerBar::PowerBar(const GUI::GameTextureManager &tex) : textureManager(
tex) {
    this->animations.reserve(POWER_FRAMES_QUANTITY);
}

void Weapon::PowerBar::setAngle(float angle, Worm::Direction d) {
    this->angle = d == Worm::Direction::right ? angle : 180 - angle;
}

void Weapon::PowerBar::update(float dt) {
    if (this->shotStarted) {
        this->elapsedTime += dt;
        if (this->power < POWER_FRAMES_QUANTITY && this->elapsedTime < POWER_CHARGE_TIME) {
            this->animations.emplace_back(this->textureManager.get(GUI::GameTextures::PowerBar),
                                           false, this->power, false);
            this->power++;
        }
    }
}

void Weapon::PowerBar::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) {
    for (int i = 0; i < this->power; i++) {
        GUI::Position powerPos =
            GUI::Position((SCOPE_DISTANCE * (log10(10 * i / 17))) * cos(this->angle * PI / 180),
                          (SCOPE_DISTANCE * (log10(10 * i / 17))) * sin(this->angle * PI / 180)) +
            p;
        this->animations[i].render(powerPos, cam, flip);
    }
}

void Weapon::PowerBar::startShot() {
    this->shotStarted = true;
}

void Weapon::PowerBar::endShot() {
    this->shotStarted = false;
    this->animations.erase(this->animations.begin(), this->animations.end());
    this->power = 0;
    this->elapsedTime = 0.0f;
}

```

jun 26, 18 2:39

PowerBar.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __PowerBar_H__
#define __PowerBar_H__

#include <Animation.h>
#include <Camera.h>
#include <vector>

#include "../GameTextures.h"
#include "Direction.h"

#define POWER_FRAMES_QUANTITY 16

namespace Weapon {
class PowerBar {
public:
    explicit PowerBar(const GUI::GameTextureManager &tex);
    ~PowerBar() = default;
    void setAngle(float angle, Worm::Direction d);
    void update(float dt);
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip);
    void startShot();
    void endShot();

private:
    bool shotStarted{false};
    float angle{0.0f};
    float elapsedTime{0.0f};
    uint16_t power{0};
    std::vector<GUI::Animation> animations;
    const GUI::GameTextureManager &textureManager;
};
}

#endif //__PowerBar_H__

```

jun 26, 18 2:39

Scope.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include "Scope.h"
#include "Direction.h"
#include "Weapon.h"

Weapon::Scope::Scope(const GUI::GameTextureManager &tex)
    : animation(tex.get(GUI::GameTextures::Scope), false, 0, false) {}

void Weapon::Scope::setAngle(float angle, Worm::Direction d) {
    this->angle = d == Worm::Direction::right ? angle : 180 - angle;
}

void Weapon::Scope::update(float dt) {
    this->animation.update(dt);
}

void Weapon::Scope::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip
&flip) {
    GUI::Position scopePos = GUI::Position(SCOPE_DISTANCE * cos(this->angle * PI
/ 180),
                                           SCOPE_DISTANCE * sin(this->angle * PI
/ 180)) +
                                           p;
    this->animation.render(scopePos, cam, flip);
}

```


jun 26, 18 2:39

Scope.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __Scope_H__
#define __Scope_H__

#include <Animation.h>
#include <Camera.h>
#include "../GameTextures.h"
#include "Direction.h"

namespace Weapon {
class Scope {
public:
    Scope(const GUI::GameTextureManager &tex);
    ~Scope() = default;
    void setAngle(float angle, Worm::Direction d);
    void update(float dt);
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip);

private:
    float angle{0.0f};
    GUI::Animation animation;
};
}

#endif //__Scope_H__
```

jun 29, 18 16:28

SelectActionWindow.cpp

Page 1/1

```

//
// Created by rodrigo on 19/06/18.
//

#include <SDL2/SDL.h>
#include <iostream>

#include "SelectActionWindow.h"
#include "Text.h"
#include "Window.h"

#define MSG_CREATE_GAME "Create game"
#define MSG_JOIN_GAME "Join game"

GUI::SelectActionWindow::SelectActionWindow(Window &window, Font &font, Camera &cam) :
    GameWindow(window, font, cam) {
    std::string msg(MSG_CREATE_GAME);
    this->buttons.emplace_back(ScreenPosition{this->window.getWidth() / 4, this->window.getHeight() / 2},
                               50, 300, msg, this->font);
    msg = MSG_JOIN_GAME;
    this->buttons.emplace_back(ScreenPosition{this->window.getWidth() * 3 / 4, this->window.getHeight() / 2},
                               50, 300, msg, this->font);
}

void GUI::SelectActionWindow::start() {
}

void GUI::SelectActionWindow::render() {
    this->window.clear(SDL_Color{0xFF, 0xFF, 0xFF});
    for (auto &button : this->buttons) {
        button.render(this->cam);
    }
    this->window.render();
}

void GUI::SelectActionWindow::buttonPressed(GUI::ScreenPosition sp) {
    if (this->buttons[0].inside(sp)) {
        this->notify(*this, Event::CreateGame);
    }

    if (this->buttons[1].inside(sp)) {
        this->notify(*this, Event::JoinGame);
    }
}

void GUI::SelectActionWindow::appendCharacter(char *text) {
}

void GUI::SelectActionWindow::handleKeyDown(SDL_Keycode key) {
}

```

jun 29, 18 16:28

SelectActionWindow.h

Page 1/1

```
//  
// Created by rodrigo on 19/06/18.  
//  
  
#ifndef INC_4_WORMS_SELECTACTIONWINDOW_H  
#define INC_4_WORMS_SELECTACTIONWINDOW_H  
  
#include <vector>  
  
#include "Window.h"  
#include "Font.h"  
#include "GameWindow.h"  
#include "Button.h"  
  
namespace GUI {  
    class SelectActionWindow : public GameWindow {  
    public:  
        explicit SelectActionWindow(Window &window, Font &font, Camera &cam);  
  
        void start() override;  
        void render() override;  
        void handleKeyDown(SDL_Keycode key) override;  
        void appendCharacter(char text[32]) override;  
        void buttonPressed(ScreenPosition sp) override;  
  
    private:  
        std::vector<Button> buttons;  
    };  
}  
  
#endif //INC_4_WORMS_SELECTACTIONWINDOW_H
```

jun 26, 18 2:39	Sliding.cpp	Page 1/2
<pre>#include "Sliding.h" Worm::Sliding::Sliding() : State(StateID::Sliding) {} Worm::Sliding::~~Sliding() {} void Worm::Sliding::update(float dt) {} IO::PlayerInput Worm::Sliding::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::mortar(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::banana(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39	Sliding.cpp	Page 2/2
<pre>} IO::PlayerInput Worm::Sliding::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Sliding::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39

Sliding.h

Page 1/1

```

#ifndef PLAYER_SLIDING_H_
#define PLAYER_SLIDING_H_

#include "WormState.h"

namespace Worm {
class Sliding : public State {
public:
    Sliding();
    ~Sliding();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;

    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
};
} // namespace Worm

#endif // INC_4_WORMS_FALLING_H

```

jun 29, 18 16:28

SoundEffect.cpp

Page 1/1

```
//  
// Created by rodrigo on 4/06/18.  
//  
  
#include "SoundEffect.h"  
#include "Exception.h"  
  
GUI::SoundEffect::SoundEffect(const std::string &filename) {  
    this->soundEffect = Mix_LoadWAV(filename.c_str());  
    if (!this->soundEffect) {  
        throw Exception{"Error loading %s: %s", filename.c_str(), Mix_GetError()};  
    }  
}  
  
GUI::SoundEffect::~SoundEffect() {  
    if (this->soundEffect != nullptr) {  
        Mix_FreeChunk(this->soundEffect);  
    }  
}  
  
Mix_Chunk *GUI::SoundEffect::getChunk() const {  
    return this->soundEffect;  
}  
  
GUI::SoundEffect::SoundEffect(GUI::SoundEffect &&other) {  
    std::swap(this->soundEffect, other.soundEffect);  
}  
  
void GUI::SoundEffect::play(bool loop) const {  
    Mix_PlayChannel(-1, this->soundEffect, -1 * loop);  
}
```

jun 29, 18 16:28

SoundEffect.h

Page 1/1

```
//  
// Created by rodrigo on 4/06/18.  
//  
  
#ifndef INC_4_WORMS_SOUNDEFFECT_H  
#define INC_4_WORMS_SOUNDEFFECT_H  
  
#include <SDL2/SDL.h>  
#include <SDL2/SDL_mixer.h>  
#include <string>  
  
namespace GUI {  
class SoundEffect {  
public:  
    SoundEffect(const std::string &filename);  
    SoundEffect(SoundEffect &&other);  
    ~SoundEffect();  
    Mix_Chunk *getChunk() const;  
    void play(bool loop) const;  
  
private:  
    Mix_Chunk *soundEffect{nullptr};  
};  
}  
  
#endif // INC_4_WORMS_SOUNDEFFECT_H
```

jun 26, 18 2:39

SoundEffectManager.h

Page 1/1

```

//
// Created by rodrigo on 4/06/18.
//

#ifndef INC_4_WORMS_SOUNDEFFECTMANAGER_H
#define INC_4_WORMS_SOUNDEFFECTMANAGER_H

#include <SDL2/SDL.h>
#include <functional>
#include <string>
#include <unordered_map>
#include "SoundEffect.h"

namespace GUI {
template <typename ID, typename HASH = std::hash<ID>>
class SoundEffectManager {
public:
    SoundEffectManager();
    ~SoundEffectManager();
    SoundEffectManager& operator=(SoundEffectManager& other) = delete;

    void load(ID id, const std::string& file_name);
    const SoundEffect& get(ID id) const;

private:
    std::unordered_map<ID, SoundEffect, HASH> cache;
};
} // namespace GUI

template <typename ID, typename HASH>
GUI::SoundEffectManager<ID, HASH>::SoundEffectManager() {}

template <typename ID, typename HASH>
GUI::SoundEffectManager<ID, HASH>::~~SoundEffectManager() {}

/**
 * @brief Loads a sound effect.
 *
 * @param file_name The image file name.
 */
template <typename ID, typename HASH>
void GUI::SoundEffectManager<ID, HASH>::load(ID id, const std::string& file_name)
{
    GUI::SoundEffect soundEffect{file_name};
    this->cache.insert(std::make_pair(id, std::move(soundEffect)));
}

/**
 * @brief Gets a sound effect.
 *
 * @param file_name Name of the sound effect.
 */
template <typename ID, typename HASH>
const GUI::SoundEffect& GUI::SoundEffectManager<ID, HASH>::get(ID id) const {
    return this->cache.at(id);
}

#endif // INC_4_WORMS_SOUNDEFFECTMANAGER_H

```


jun 29, 18 16:28

SoundEffectPlayer.cpp

Page 1/1

```

//
// Created by rodrigo on 5/06/18.
//

#include "SoundEffectPlayer.h"

GUI::SoundEffectPlayer::SoundEffectPlayer(const GUI::SoundEffect &soundEffect)
    : soundEffect(&soundEffect) {}

GUI::SoundEffectPlayer::SoundEffectPlayer(const GUI::SoundEffect &soundEffect, float duration)
    : soundEffect(&soundEffect), duration(duration) {
    // this->soundEffect->play();
}

GUI::SoundEffectPlayer::SoundEffectPlayer(const GUI::SoundEffect &soundEffect, bool autoUpdate)
    : soundEffect(&soundEffect), autoUpdate(autoUpdate) {}

GUI::SoundEffectPlayer::~SoundEffectPlayer() {}

void GUI::SoundEffectPlayer::update(float dt) {
    if (!this->autoUpdate) {
        this->timeElapsed += dt;
        if (this->timeElapsed > this->duration) {
            this->play();
            this->timeElapsed = 0.0f;
        }
    }
}

void GUI::SoundEffectPlayer::play() {
    this->soundEffect->play(this->loop);
}

```

jun 29, 18 16:28

SoundEffectPlayer.h

Page 1/1

```
//  
// Created by rodrigo on 5/06/18.  
//  
  
#ifndef INC_4_WORMS_SOUNDEFFECTPLAYER_H  
#define INC_4_WORMS_SOUNDEFFECTPLAYER_H  
  
#include <SDL2/SDL.h>  
  
#include "SoundEffect.h"  
  
namespace GUI {  
class SoundEffectPlayer {  
public:  
    bool loop{false};  
  
    explicit SoundEffectPlayer(const GUI::SoundEffect &soundEffect);  
    SoundEffectPlayer(const SoundEffect &soundEffect, float duration);  
    SoundEffectPlayer(const GUI::SoundEffect &soundEffect, bool autoUpdate);  
    ~SoundEffectPlayer();  
    void update(float dt);  
    void play();  
  
private:  
    const SoundEffect *soundEffect;  
    float duration{0.0f};  
    float timeElapsed{0.0f};  
    bool autoUpdate{false};  
};  
}  
  
#endif // INC_4_WORMS_SOUNDEFFECTPLAYER_H
```

jun 26, 18 2:39

Teleport.cpp

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#include "Teleport.h"

Worm::Teleport::Teleport(const GUI::GameTextureManager &tex)
    : Weapon(tex, GUI::GameTextures::WormTeleport, TELEPORT_CENTER_FRAME, Weapon
ID::WTeleport) {
    this->weaponAnimation.setAnimateOnce();
}

void Worm::Teleport::update(float dt) {
    if (!this->weaponAnimation.finished()) {
        this->weaponAnimation.update(dt);
    } else {
        this->endAnimation();
    }
}

void Worm::Teleport::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip
&flip) {
    this->weaponAnimation.render(p, cam, flip);
}

void Worm::Teleport::setAngle(float angle, Worm::Direction d) {}

void Worm::Teleport::startShot() {}

void Worm::Teleport::endShot() {}

bool Worm::Teleport::positionSelected() {
    this->weaponAnimation.setAutoUpdate(true);
    return true;
}

void Worm::Teleport::endAnimation() {
    this->weaponAnimation.setFrame(TELEPORT_CENTER_FRAME);
    this->weaponAnimation.setAutoUpdate(false);
}

```

jun 26, 18 2:39	Teleported.cpp	Page 1/2
<pre>// // Created by rodrigo on 16/06/18. // #include "Teleported.h" Worm::Teleported::Teleported() : State(StateID::Teleported) {} Worm::Teleported::~Teleported() {} void Worm::Teleported::update(float dt) {} IO::PlayerInput Worm::Teleported::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::mortar(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39	Teleported.cpp	Page 2/2
<pre>} IO::PlayerInput Worm::Teleported::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleported::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39

Teleported.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_TELEPORTED_H
#define INC_4_WORMS_TELEPORTED_H

#include "WormState.h"

namespace Worm {
class Teleported : public State {
public:
    Teleported();
    ~Teleported();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // INC_4_WORMS_TELEPORTED_H

```

jun 26, 18 2:39

Teleport.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_TELEPORT_H
#define INC_4_WORMS_TELEPORT_H

#define TELEPORT_CENTER_FRAME 0

#include "Weapon.h"

namespace Worm {
class Teleport : public Weapon {
public:
    explicit Teleport(const GUI::GameTextureManager &textureManager);
    ~Teleport() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;

private:
    void endAnimation();
};
} // namespace Worm

#endif // INC_4_WORMS_TELEPORT_H

```

jun 26, 18 2:39	Teleporting.cpp	Page 1/2
<pre>// // Created by rodrigo on 16/06/18. // #include "Teleporting.h" Worm::Teleporting::Teleporting() : State(StateID::Teleporting) {} Worm::Teleporting::~Teleporting() {} void Worm::Teleporting::update(float dt) {} IO::PlayerInput Worm::Teleporting::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::mortar(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39	Teleporting.cpp	Page 2/2
<pre>} IO::PlayerInput Worm::Teleporting::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Teleporting::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; }</pre>		

jun 26, 18 2:39

Teleporting.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_TELEPORTING_H
#define INC_4_WORMS_TELEPORTING_H

#include "WormState.h"

namespace Worm {
class Teleporting : public State {
public:
    Teleporting();
    ~Teleporting();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // INC_4_WORMS_TELEPORTING_H

```


jun 29, 18 16:28

WaitingPlayersWindow.cpp

Page 1/1

```

//
// Created by rodrigo on 24/06/18.
//

#include "WaitingPlayersWindow.h"

GUI::WaitingPlayersWindow::WaitingPlayersWindow(GUI::Window &window, GUI::Font &
font, GUI::Camera &cam,
                                uint8_t playersQuantity) :
    GameWindow(window, font, cam),
    playersQuantity(playersQuantity) {
}

GUI::WaitingPlayersWindow::WaitingPlayersWindow(GUI::Window &window, GUI::Font &
font, GUI::Camera &cam,
                                uint8_t playersQuantity, uint8_t
playersConnected) :
    WaitingPlayersWindow(window, font, cam, playersQuantity) {
    this->playersConnected = playersConnected;
}

void GUI::WaitingPlayersWindow::start() {
}

void GUI::WaitingPlayersWindow::render() {
    this->window.clear(SDL_Color{0xFF, 0xFF, 0xFF});

    Text playersConnected(this->font);
    int x = this->window.getWidth() * 2 / 5;
    int y = this->window.getHeight() / 2;
    playersConnected.set("Players connected", SDL_Color{0, 0, 0}, 50);
    playersConnected.renderFixed(ScreenPosition{x, y}, this->cam);
    x = this->window.getWidth() * 3 / 5;
    y = this->window.getHeight() / 2;
    playersConnected.setBackground(SDL_Color{0, 0, 0});
    playersConnected.set(std::to_string(this->playersConnected) + "/" + std::to_
string(this->playersQuantity), SDL_Color{0xFF, 0xFF, 0xFF}, 50);
    playersConnected.renderFixed(ScreenPosition{x, y}, this->cam);

    this->window.render();
}

void GUI::WaitingPlayersWindow::buttonPressed(GUI::ScreenPosition sp) {
}

void GUI::WaitingPlayersWindow::appendCharacter(char *text) {
}

void GUI::WaitingPlayersWindow::handleKeyDown(SDL_Keycode key) {
}

```

jun 29, 18 16:28

WaitingPlayersWindow.h

Page 1/1

```

//
// Created by rodrigo on 24/06/18.
//

#ifndef INC_4_WORMS_WAITINGPLAYERSWINDOW_H
#define INC_4_WORMS_WAITINGPLAYERSWINDOW_H

#include <vector>

#include "Window.h"
#include "Font.h"
#include "GameStateMsg.h"
#include "GameWindow.h"
#include "Button.h"

namespace GUI {
    class WaitingPlayersWindow : public GameWindow {
    public:
        uint8_t playersConnected{0};

        WaitingPlayersWindow(GUI::Window &window, GUI::Font &font, GUI::Camera &
cam, uint8_t playersQuantity);
        WaitingPlayersWindow(Window &window, Font &font, Camera &cam, uint8_t pl
ayersQuantity, uint8_t playersConnected);

        void start() override;
        void render() override;
        void handleKeyDown(SDL_Keycode key) override;
        void appendCharacter(char text[32]) override;
        void buttonPressed(ScreenPosition sp) override;

    private:
        std::vector<Button> buttons;
        unsigned int playersQuantity{0};
    };
}

#endif //INC_4_WORMS_WAITINGPLAYERSWINDOW_H

```

jun 26, 18 7:40

Water.cpp

Page 1/1

```

#include "Water.h"
#include <cmath>
#include "WrapTexture.h"

GUI::Water::Water(const GUI::GameTextureManager &tm) : textureManager(tm) {}

/**
 * @brief Updates the water animation state
 *
 * @param dt Time elapsed since the last call to this function.
 */
void GUI::Water::update(float dt) {
    this->elapsed += dt;
    this->yDelta = std::sin(this->elapsed) * 1;
}

/**
 * @brief Renders the water.
 *
 * @param camera Camera where the water is rendered.
 */
void GUI::Water::render(GUI::Camera &camera) {
    const GUI::Texture &texture = this->textureManager.get(GUI::GameTextures::Water);
    GUI::WrapTexture water{texture, camera.screenWidth(), texture.getHeight() / camera.getScale()};
    water.render(Position{camera.getPosition().x, -6.5f + this->yDelta}, camera);
}

```

jun 26, 18 7:40

Water.h

Page 1/1

```
#ifndef WATER_H_
#define WATER_H_

#include "Camera.h"
#include "GameTextures.h"

namespace GUI {
class Water {
public:
    Water(const GameTextureManager &tm);
    ~Water() = default;

    void update(float dt);
    void render(Camera &camera);

private:
    const GUI::GameTextureManager &textureManager;
    float elapsed{0};
    float yDelta{0};
};
} // namespace GUI

#endif
```

jun 26, 18 2:39

Weapon.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 27/05/18
 */

#include <iostream>

#include "GameStateMsg.h"
#include "Weapon.h"

Worm::Weapon::Weapon(const GUI::GameTextureManager &texMgr, GUI::GameTextures texMgr,
                    uint16_t centerFrame, WeaponID id)
    : textureMgr(texMgr),
      current(id),
      centerFrame(centerFrame),
      weaponAnimation(texMgr.get(tex), false, centerFrame, false) {}

const Worm::WeaponID &Worm::Weapon::getWeaponID() const {
    return this->current;
}
```

jun 26, 18 2:39	Weapon.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 27/05/18 */ #ifndef __Weapon_H__ #define __Weapon_H__ #include "../GameTextures.h" #include "Animation.h" #include "Camera.h" #include "Direction.h" #include "GameStateMsg.h" #include "TextureManager.h" #define ANGLE_STEP 5.625f #define SCOPE_DISTANCE 4 namespace Worm { class Weapon { public: explicit Weapon(const GUI::GameTextureManager &texMgr, GUI::GameTextures tex , uint16_t centerFrame, WeaponID id); virtual ~Weapon() = default; /** * updates all its animations. * @param dt */ virtual void update(float dt) = 0; /** * renders all its animations. * @param p * @param cam * @param flip */ virtual void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) = 0; const WeaponID &getWeaponID() const; /** * updates animations' frame depending on the angle. * @param angle */ virtual void setAngle(float angle, Direction d) = 0; /** * Starts the PowerBar's rendering, adding animations in its container */ virtual void startShot() = 0; /** * End PowerBar's rendering, freeing its container */ virtual void endShot() = 0; /** * When using remoteControl weapons, starts the animation of the worm * and return */ virtual bool positionSelected() = 0; protected: const GUI::GameTextureManager &textureMgr; WeaponID current; uint16_t centerFrame; </pre>		

jun 26, 18 2:39	Weapon.h	Page 2/2
<pre> GUI::Animation weaponAnimation; float angle{0.0f}; }; } // namespace Weapon #endif // __Weapon_H__ </pre>		

jun 26, 18 2:39

WeaponNone.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#include "WeaponNone.h"

Worm::WeaponNone::WeaponNone(const GUI::GameTextureManager &textureManager)
    : Weapon(textureManager, GUI::GameTextures::WormIdle, 0, WeaponID::WNone) {}

void Worm::WeaponNone::update(float dt) {}

void Worm::WeaponNone::render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) {}

void Worm::WeaponNone::setAngle(float angle, Worm::Direction d) {}

void Worm::WeaponNone::startShot() {}

void Worm::WeaponNone::endShot() {}

bool Worm::WeaponNone::positionSelected() {
    return false;
}
```

jun 26, 18 2:39

WeaponNone.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __WEAPON_NONE_H__
#define __WEAPON_NONE_H__

#include <vector>

#include "Weapon.h"

namespace Worm {
class WeaponNone : public Weapon {
public:
    explicit WeaponNone(const GUI::GameTextureManager &textureManager);
    ~WeaponNone() = default;
    void update(float dt) override;
    void render(GUI::Position &p, GUI::Camera &cam, SDL_RendererFlip &flip) override;
    void setAngle(float angle, Direction d) override;
    void startShot() override;
    void endShot() override;
    bool positionSelected() override;
};
} // namespace Worm

#endif //__WEAPON_NONE_H__

```


jun 26, 18 2:39

Wind.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/06/18
 */

#include "Wind.h"
#include <WrapTexture.h>
#include "Texture.h"

GUI::Wind::Wind(const GUI::GameTextureManager &tex, GUI::Camera &cam) : tex(tex)
, cam(cam) {}

void GUI::Wind::render(std::int8_t intensity, int windowHeight) {
    const GUI::Texture &toUse = (intensity > 0) ? this->tex.get(GameTextures::Wi
ndRight)
                                : this->tex.get(GameTextures::Wi
ndLeft);
    float scaledIntensity = (float)std::abs(intensity) / 127 * this->cam.getScal
e();
    GUI::WrapTexture wt{toUse, scaledIntensity, (float)toUse.getHeight() / this-
>cam.getScale()};
    GUI::ScreenPosition p{windowWidth, toUse.getHeight()};
    wt.renderFixed(p, this->cam);
}

```

jun 26, 18 2:39

Wind.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/06/18
 */

#ifndef __WIND_H__
#define __WIND_H__

#include <Camera.h>
#include "GameTextures.h"

namespace GUI {
/**
 * @brief receives the snapshot's intensity and draws the help interface
 * to show the wind's intensity.
 */
class Wind {
public:
    Wind(const GameTextureManager &textureManager, Camera &cam);
    ~Wind() = default;
    void render(std::int8_t intensity, int windowHeight);

private:
    const GameTextureManager &tex;
    Camera &cam;
};
}

#endif //__WIND_H__
```

jun 26, 18 2:39	WormBackFlipping.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 21/05/18 */ #include "WormBackFlipping.h" Worm::BackFlipping::BackFlipping() : State(StateID::BackFlipping) {} Worm::BackFlipping::~BackFlipping() {} void Worm::BackFlipping::update(float dt) {} IO::PlayerInput Worm::BackFlipping::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	WormBackFlipping.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::BackFlipping::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

WormBackFlipping.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#ifndef __WORM_BACK_FLIPPING_H__
#define __WORM_BACK_FLIPPING_H__

#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class BackFlipping : public State {
public:
    BackFlipping();
    ~BackFlipping();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;

    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __WORM_BACK_FLIPPING_H__

```

jun 26, 18 7:40

Worm.cpp

Page 1/10

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#include <SDL2/SDL_system.h>
#include <cmath>
#include <iostream>

#include "GameStateMsg.h"
#include "Text.h"
#include "Weapons/AerialAttack.h"
#include "Weapons/Banana.h"
#include "Weapons/BaseballBat.h"
#include "Weapons/Bazooka.h"
#include "Weapons/Cluster.h"
#include "Weapons/Dynamite.h"
#include "Weapons/Grenade.h"
#include "Weapons/Holy.h"
#include "Weapons/Mortar.h"
#include "Weapons/Teleport.h"
#include "Weapons/WeaponNone.h"
#include "Worm.h"
#include "WormState/BackFlip.h"
#include "WormState/Batting.h"
#include "WormState/Dead.h"
#include "WormState/Die.h"
#include "WormState/Drowning.h"
#include "WormState/Falling.h"
#include "WormState/Hit.h"
#include "WormState/Land.h"
#include "WormState/Sliding.h"
#include "WormState/Teleported.h"
#include "WormState/Teleporting.h"
#include "WormState/WormBackFlipping.h"
#include "WormState/WormEndBackFlip.h"
#include "WormState/WormEndJump.h"
#include "WormState/WormJumping.h"
#include "WormState/WormStartJump.h"
#include "WormState/WormStill.h"
#include "WormState/WormWalk.h"

Worm::Worm::Worm(ID id, const GUI::GameTextureManager &texture_mgr,
                 const GUI::GameSoundEffectManager &sound_effect_mgr)
    : id(id),
      texture_mgr(texture_mgr),
      sound_effect_mgr(sound_effect_mgr),
      animation(texture_mgr.get(GUI::GameTextures::WormIdle)) {
    this->setState(Worm::StateID::Still);
    this->weapon = std::shared_ptr<Weapon>(new Bazooka(texture_mgr));
}

void Worm::Worm::handleKeyDown(SDL_Keycode key, IO::Stream<IO::PlayerMsg> *out)
{
    IO::PlayerInput i = IO::PlayerInput::moveNone;
    switch (key) {
        case SDLK_RIGHT:
            i = this->state->moveRight(*this);
            break;
        case SDLK_LEFT:
            i = this->state->moveLeft(*this);
            break;
    }
}

```

jun 26, 18 7:40

Worm.cpp

Page 2/10

```

case SDLK_UP:
    i = this->state->pointUp(*this);
    break;
case SDLK_DOWN:
    i = this->state->pointDown(*this);
    break;
case SDLK_RETURN:
    i = this->state->jump(*this);
    break;
case SDLK_BACKSPACE:
    i = this->state->backFlip(*this);
    break;
case SDLK_1:
    i = this->state->setTimeoutTo(*this, 1);
    break;
case SDLK_2:
    i = this->state->setTimeoutTo(*this, 2);
    break;
case SDLK_3:
    i = this->state->setTimeoutTo(*this, 3);
    break;
case SDLK_4:
    i = this->state->setTimeoutTo(*this, 4);
    break;
case SDLK_5:
    i = this->state->setTimeoutTo(*this, 5);
    break;
case SDLK_F1:
    i = this->state->bazooka(*this);
    break;
case SDLK_F2:
    i = this->state->grenade(*this);
    break;
case SDLK_F3:
    i = this->state->cluster(*this);
    break;
case SDLK_F4:
    i = this->state->mortar(*this);
    break;
case SDLK_F5:
    i = this->state->banana(*this);
    break;
case SDLK_F6:
    i = this->state->holy(*this);
    break;
case SDLK_F7:
    i = this->state->aerialAttack(*this);
    break;
case SDLK_F8:
    i = this->state->dynamite(*this);
    break;
case SDLK_F9:
    i = this->state->baseballBat(*this);
    break;
case SDLK_F10:
    i = this->state->teleport(*this);
    break;
case SDLK_SPACE:
    i = this->state->startShot(*this);
    break;
}
if (i != IO::PlayerInput::moveNone) {

```

jun 26, 18 7:40

Worm.cpp

Page 3/10

```

        IO::PlayerMsg msg;
        msg.input = i;
        *out << msg;
    }
}

void Worm::Worm::handleKeyUp(SDL_Keycode key, IO::Stream<IO::PlayerMsg> *out) {
    IO::PlayerInput i = IO::PlayerInput::moveNone;
    switch (key) {
        case SDLK_RIGHT:
            i = this->state->stopMove(*this);
            break;
        case SDLK_LEFT:
            i = this->state->stopMove(*this);
            break;
        case SDLK_SPACE:
            i = this->state->endShot(*this);
            break;
    }
    if (i != IO::PlayerInput::moveNone) {
        IO::PlayerMsg msg;
        msg.input = i;
        *out << msg;
    }
}

void Worm::Worm::render(GUI::Position &p, GUI::Camera &cam) {
    SDL_RendererFlip flipType =
        this->direction == Direction::left ? SDL_FLIP_NONE : SDL_FLIP_HORIZONTAL;
    if (this->state->getState() != StateID::Still ||
        this->weapon->getWeaponID() == WeaponID::WNone) {
        this->animation.render(p, cam, flipType);
    } else {
        this->weapon->render(p, cam, flipType);
    }
    if (this->explosion != nullptr) {
        this->explosion->render(cam);
        if (this->explosion->finished()) {
            this->explosion = nullptr;
        }
    }
}

void Worm::Worm::update(float dt) {
    this->state->update(dt);
    this->animation.update(dt);
    this->weapon->update(dt);
    if (this->explosion != nullptr) {
        this->explosion->update(dt);
    }
    if (this->soundEffectPlayer != nullptr) {
        this->soundEffectPlayer->update(dt);
    }
}

GUI::Animation Worm::Worm::getAnimation(StateID state) const {
    switch (state) {
        case StateID::Still:
            break;
        case StateID::Walk:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::WormW

```

jun 26, 18 7:40

Worm.cpp

Page 4/10

```

alk));
        case StateID::StartBackFlip:
        case StateID::StartJump:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::Start
Jump), true};
        case StateID::Jumping:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::Jumpi
ng)};
        case StateID::Land:
        case StateID::EndBackFlip:
        case StateID::EndJump:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::EndJu
mp), true};
        case StateID::BackFlipping: {
            GUI::Animation animation{this->texture_mgr.get(GUI::GameTextures::Ba
ckFlipping)};
            animation.setAnimateOnce();
            return animation;
        }
        case StateID::Falling: {
            GUI::Animation animation{this->texture_mgr.get(GUI::GameTextures::Fa
lling), true};
            animation.setAnimateOnce();
            return animation;
        }
        case StateID::Batting: {
            GUI::Animation animation{this->texture_mgr.get(GUI::GameTextures::Wo
rmBaseballBatting),
                false, 25, false};
            // animation.setAnimateOnce();
            return animation;
        }
        case StateID::Teleporting: {
            GUI::Animation animation{this->texture_mgr.get(GUI::GameTextures::Wo
rmTeleporting),
                true};
            animation.setAnimateOnce();
            return animation;
        }
        case StateID::Teleported: {
            GUI::Animation animation{this->texture_mgr.get(GUI::GameTextures::Wo
rmTeleporting),
                true};
            animation.setPlayInverse();
            return animation;
        }
        case StateID::Hit:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::Fly),
                true,
                FLY_CENTER_FRAME, false};
        case StateID::Die: {
            GUI::Animation animation{this->texture_mgr.get(GUI::GameTextures::Di
e)};
            animation.setAnimateOnce();
            return animation;
        }
        case StateID::Drowning:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::Fly),
                true,
                DROWN_CENTER_FRAME, false};
        case StateID::Dead:
            return GUI::Animation{this->texture_mgr.get(GUI::GameTextures::Dead)

```

jun 26, 18 7:40	Worm.cpp	Page 5/10
<pre> , true); case StateID::Sliding: return GUI::Animation(this->texture_mgr.get(GUI::GameTextures::Sliding), true); } return GUI::Animation(this->texture_mgr.get(GUI::GameTextures::WormIdle), true); } void Worm::Worm::playSoundEffect(StateID state) { this->soundEffectPlayer = nullptr; switch (state) { case StateID::Still: break; case StateID::Walk: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WalkCompress), 0.7f)); this->soundEffectPlayer->update(0.3f); break; case StateID::StartBackFlip: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WormBackFlip), true)); this->soundEffectPlayer->play(); break; case StateID::StartJump: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WormJump), true)); this->soundEffectPlayer->play(); break; case StateID::Jumping: break; case StateID::EndBackFlip: case StateID::EndJump: case StateID::Land: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WormLanding), true)); this->soundEffectPlayer->play(); break; case StateID::BackFlipping: break; case StateID::Falling: break; case StateID::Batting: break; case StateID::Teleporting: break; case StateID::Teleported: break; case StateID::Hit: this->soundEffectPlayer = </pre>		

jun 26, 18 7:40	Worm.cpp	Page 6/10
<pre> std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WormHit), true)); this->soundEffectPlayer->play(); break; case StateID::Die: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WormDie), true)); this->soundEffectPlayer->play(); break; case StateID::Drowning: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::WormDrowning))); break; case StateID::Dead: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer(this->sound_effect_mgr.get(GUI::GameSoundEffects::Explosion), true)); this->soundEffectPlayer->play(); break; case StateID::Sliding: break; } } void Worm::Worm::setState(StateID state) { if (this->state == nullptr this->state->getState() != state) { this->animation = this->getAnimation(state); this->playSoundEffect(state); /* creates the right state type */ switch (state) { case StateID::Still: this->state = std::shared_ptr<State>(new Still()); break; case StateID::Walk: this->state = std::shared_ptr<State>(new Walk()); break; case StateID::StartJump: this->state = std::shared_ptr<State>(new StartJump()); break; case StateID::Jumping: this->state = std::shared_ptr<State>(new Jumping()); break; case StateID::EndJump: this->state = std::shared_ptr<State>(new EndJump()); break; case StateID::StartBackFlip: this->state = std::shared_ptr<State>(new BackFlip()); break; case StateID::BackFlipping: this->state = std::shared_ptr<State>(new BackFlipping()); break; </pre>		

jun 26, 18 7:40

Worm.cpp

Page 7/10

```

        case StateID::EndBackFlip:
            this->state = std::shared_ptr<State>(new EndBackFlip());
            break;
        case StateID::Falling:
            this->state = std::shared_ptr<State>(new Falling());
            break;
        case StateID::Land:
            this->state = std::shared_ptr<State>(new Land());
            break;
        case StateID::Batting:
            this->state = std::shared_ptr<State>(new Batting());
            break;
        case StateID::Teleporting:
            this->state = std::shared_ptr<State>(new Teleporting());
            break;
        case StateID::Teleported:
            this->state = std::shared_ptr<State>(new Teleported());
            break;
        case StateID::Hit:
            this->state = std::shared_ptr<State>(new Hit());
            break;
        case StateID::Die:
            this->state = std::shared_ptr<State>(new Die());
            break;
        case StateID::Drowning:
            this->state = std::shared_ptr<State>(new Drowning());
            break;
        case StateID::Dead:
            this->state = std::shared_ptr<State>(new Dead());
            this->explosion = std::shared_ptr<Explosion>(new Explosion(this-
>texture_mgr));
            this->explosion->position = this->position;
            break;
        case StateID::Sliding:
            this->state = std::shared_ptr<State>(new Sliding());
            break;
    }
}

Worm::StateID &Worm::Worm::getState() const {
    return this->state->getState();
}

void Worm::Worm::setWeapon(const WeaponID &id) {
    // this->weapon.setWeapon(id);
    if (this->weapon->getWeaponID() != id) {
        switch (id) {
            case WeaponID::WBazooka:
                this->weapon = std::shared_ptr<Weapon>(new Bazooka(this->texture
_mgr));
                break;
            case WeaponID::WGrenade:
                this->weapon = std::shared_ptr<Weapon>(new Grenade(this->texture
_mgr));
                break;
            case WeaponID::WCluster:
                this->weapon = std::shared_ptr<Weapon>(new Cluster(this->texture
_mgr));
                break;
            case WeaponID::WMortar:
                this->weapon = std::shared_ptr<Weapon>(new Mortar(this->texture_

```

jun 26, 18 7:40

Worm.cpp

Page 8/10

```

_mgr));
                break;
            case WeaponID::WBanana:
                this->weapon = std::shared_ptr<Weapon>(new Banana(this->texture_
_mgr));
                break;
            case WeaponID::WHoly:
                this->weapon = std::shared_ptr<Weapon>(new Holy(this->texture_mg
r));
                break;
            case WeaponID::WAerial:
                this->weapon = std::shared_ptr<Weapon>(new AerialAttack(this->te
xture_mgr));
                break;
            case WeaponID::WDynamite:
                this->weapon = std::shared_ptr<Weapon>(new Dynamite(this->textur
e_mgr));
                break;
            case WeaponID::WBaseballBat:
                this->weapon = std::shared_ptr<Weapon>(new BaseballBat(this->tex
ture_mgr));
                break;
            case WeaponID::WTeleport:
                this->weapon = std::shared_ptr<Weapon>(new Teleport(this->textur
e_mgr));
                break;
            case WeaponID::WNone:
                this->weapon = std::shared_ptr<Weapon>(new WeaponNone(this->text
ure_mgr));
                break;
            case WeaponID::WExplode:
                break;
            case WeaponID::WFragment:
                break;
        }
    }
}

const Worm::WeaponID &Worm::Worm::getWeaponID() const {
    return this->weapon->getWeaponID();
}

void Worm::Worm::setWeaponAngle(float angle) {
    this->weapon->setAngle(angle, this->direction);
}

void Worm::Worm::setPosition(GUI::Position p) {
    this->position = p;
}

void Worm::Worm::startShot() {
    if (!this->hasFired) {
        this->weapon->startShot();
    }
}

void Worm::Worm::endShot() {
    if (this->weapon->getWeaponID() != WeaponID::WAerial &&
        this->weapon->getWeaponID() != WeaponID::WTeleport &&
        this->weapon->getWeaponID() != WeaponID::WNone) {
        if (!this->hasFired) {
            this->weapon->endShot();

```


jun 26, 18 7:40	Worm.cpp	Page 9/10
<pre> this->playWeaponSoundEffect(this->getWeaponID()); this->hasFired = true; } } void Worm::Worm::mouseButtonDown(GUI::Position position, IO::Stream<IO::PlayerMsg> *out) { IO::PlayerInput i = this->state->positionSelected(*this); if (i != IO::PlayerInput::moveNone && !this->hasFired && this->weapon->positionSelected()) { this->playWeaponSoundEffect(this->weapon->getWeaponID()); IO::PlayerMsg msg; msg.input = i; msg.position = position; *out << msg; } } void Worm::Worm::playWeaponSoundEffect(const WeaponID &id) { this->soundEffectPlayer = nullptr; switch (id) { case WeaponID::WBazooka: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Shot), true e}); this->soundEffectPlayer->play(); break; case WeaponID::WGrenade: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Shot), true e}); this->soundEffectPlayer->play(); break; case WeaponID::WCluster: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Shot), true e}); this->soundEffectPlayer->play(); break; case WeaponID::WMortar: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Shot), true e}); this->soundEffectPlayer->play(); break; case WeaponID::WBanana: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Shot), true e}); this->soundEffectPlayer->play(); break; } } void Worm::Worm::reset() { this->hasFired = false; } </pre>		

jun 26, 18 7:40	Worm.cpp	Page 10/10
<pre> break; case WeaponID::WHoly: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::Holy), true e}); this->soundEffectPlayer->play(); break; case WeaponID::WAerial: this->soundEffectPlayer = std::shared_ptr<GUI::SoundEffectPlayer>(new GUI::SoundEffectPlayer{ er{ this->sound_effect_mgr.get(GUI::GameSoundEffects::AirStrike) , true}); this->soundEffectPlayer->play(); break; case WeaponID::WDynamite: break; case WeaponID::WBaseballBat: break; case WeaponID::WTeleport: break; case WeaponID::WNone: break; case WeaponID::WExplode: break; case WeaponID::WFragment: break; } } void Worm::Worm::reset() { this->hasFired = false; } </pre>		

jun 26, 18 2:39	WormEndBackFlip.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 21/05/18 */ #include "WormEndBackFlip.h" Worm::EndBackFlip::EndBackFlip() : State(StateID::EndBackFlip) {} Worm::EndBackFlip::~EndBackFlip() {} void Worm::EndBackFlip::update(float dt) {} IO::PlayerInput Worm::EndBackFlip::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	WormEndBackFlip.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndBackFlip::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

WormEndBackFlip.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#ifndef __WORM_END_BACKFLIP_H__
#define __WORM_END_BACKFLIP_H__

#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class EndBackFlip : public State {
public:
    EndBackFlip();
    ~EndBackFlip();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __WORM_END_BACKFLIP_H__

```

jun 26, 18 2:39	WormEndJump.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 21/05/18 */ #include "WormEndJump.h" Worm::EndJump::EndJump() : State(StateID::EndJump) {} Worm::EndJump::~~EndJump() {} void Worm::EndJump::update(float dt) {} IO::PlayerInput Worm::EndJump::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	WormEndJump.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::EndJump::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

WormEndJump.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#ifndef __END_JUMP_H__
#define __END_JUMP_H__

#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class EndJump : public State {
public:
    EndJump();
    ~EndJump();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __END_JUMP_H__

```

jun 26, 18 7:40	Worm.h	Page 1/3
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #ifndef __Worm_H__ #define __Worm_H__ #define FLY_CENTER_FRAME 16 #define DROWN_CENTER_FRAME 0 #define ANGLE_STEP 5.625f #include <SDL2/SDL.h> #include <memory> #include "Animation.h" #include "Camera.h" #include "Direction.h" #include "GameSoundEffects.h" #include "GameStateMsg.h" #include "GameTextures.h" #include "SoundEffectPlayer.h" #include "Stream.h" #include "Weapons/Explosion.h" #include "Weapons/Weapon.h" #include "WormState/WormState.h" #include "utils.h" namespace Worm { using ID = char; class Worm { /** * Fundamental class of the game, it is in charge of handling the user's * entries, and delegate in their attributes the rendering and animation */ public: Direction direction{Direction::left}; std::uint8_t health{0}; const ID id; explicit Worm(ID id, const GUI::GameTextureManager &texture_mgr, const GUI::GameSoundEffectManager &sound_effect_mgr); ~Worm() {} /** * @brief Calls State::update to change frame of animation * @param dt */ void update(float dt); /** * Render worm in position (x,y) * @param x * @param y */ void render(GUI::Position &p, GUI::Camera &cam); /** * @brief Using a state pattern, change its state depending on the input, an * sends it to the server * @param key * @param out */ </pre>		

jun 26, 18 7:40	Worm.h	Page 2/3
<pre> void handleKeyDown(SDL_Keycode key, IO::Stream<IO::PlayerMsg> *out); /** * @brief Same as handleKeyDown, but stops its current status. * @param key * @param out */ void handleKeyUp(SDL_Keycode key, IO::Stream<IO::PlayerMsg> *out); /** * @brief Receives a position in global coordinates and sends it to the stat e * so it can handle it. * @param position */ void mouseButtonDown(GUI::Position position, IO::Stream<IO::PlayerMsg> *pStr eam); GUI::Animation getAnimation(StateID state) const; /** * @brief Attribute that implements state pattern to change the behavior * of the class polymorphically. */ void setState(StateID state); StateID &getState() const; /** * @brief Update the animation with weapons, depending on the * worm's angle. * @param angle */ void setWeaponAngle(float angle); /** * @brief Update the used weapon * @param id */ void setWeapon(const WeaponID &id); const WeaponID &getWeaponID() const; void setPosition(GUI::Position p); /** * @brief Starts the PowerBar's rendering, adding animations in its containe r */ void startShot(); /** * @brief End PowerBar's rendering, freeing its container */ void endShot(); /** * @brief resets some attributes when the turn ends */ void reset(); private: const GUI::GameTextureManager &texture_mgr; const GUI::GameSoundEffectManager &sound_effect_mgr; std::shared_ptr<State> state{nullptr}; GUI::Animation animation; std::shared_ptr<Weapon> weapon{nullptr}; bool active{false}; GUI::Position position{0, 0}; std::shared_ptr<Explosion> explosion{nullptr}; bool hasFired{false}; std::shared_ptr<GUI::SoundEffectPlayer> soundEffectPlayer{nullptr}; void playSoundEffect(StateID state); void playWeaponSoundEffect(const WeaponID &id); </pre>		

jun 26, 18 7:40

Worm.h

Page 3/3

```
};  
} // namespace Worm  
  
#endif // __Worm__H__
```

jun 26, 18 2:39	WormJumping.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 21/05/18 */ #include "WormJumping.h" Worm::Jumping::Jumping() : State(StateID::Jumping) {} Worm::Jumping::~~Jumping() {} void Worm::Jumping::update(float dt) {} IO::PlayerInput Worm::Jumping::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	WormJumping.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Jumping::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

WormJumping.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#ifndef __JUMPING_H__
#define __JUMPING_H__

#include "../Worm.h"
#include "GameStateMsg.h"

namespace Worm {
class Jumping : public State {
public:
    explicit Jumping();
    virtual ~Jumping();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __JUMPING_H__

```

jun 26, 18 2:39	WormStartJump.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 19/05/18 */ #include "WormStartJump.h" Worm::StartJump::StartJump() : State(StateID::StartJump) {} Worm::StartJump::~~StartJump() {} void Worm::StartJump::update(float dt) {} IO::PlayerInput Worm::StartJump::moveRight(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::moveLeft(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::stopMove(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::jump(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::backFlip(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::grenade(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::mortar(Worm &w) { </pre>		

jun 26, 18 2:39	WormStartJump.cpp	Page 2/2
<pre> return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::StartJump::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

WormStartJump.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 19/05/18
 */

#ifndef __WORM_START_JUMP_H__
#define __WORM_START_JUMP_H__

#include "../Worm.h"
#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class StartJump : public State {
public:
    StartJump();
    ~StartJump();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
};
} // namespace Worm

#endif // __WORM_START_JUMP_H__

```

jun 26, 18 2:39

WormState.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#ifndef __WORM_STATE_H__
#define __WORM_STATE_H__

#include "Animation.h"
#include "GameStateMsg.h"

namespace Worm {

class Worm;

/**
 * Worm status interface. It is used to implement the state pattern and
 * thus obtain a polymorphic behavior and at the same time treat the
 * animation as a state machine
 */
class State {
public:
    State(StateID stateID) : stateID(stateID){};
    virtual ~State() = default;

    virtual void update(float dt) = 0;

    virtual IO::PlayerInput moveRight(Worm &w) = 0;
    virtual IO::PlayerInput moveLeft(Worm &w) = 0;
    virtual IO::PlayerInput stopMove(Worm &w) = 0;
    virtual IO::PlayerInput pointUp(Worm &w) = 0;
    virtual IO::PlayerInput pointDown(Worm &w) = 0;
    virtual IO::PlayerInput jump(Worm &w) = 0;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) = 0;

    virtual IO::PlayerInput bazooka(Worm &w) = 0;
    virtual IO::PlayerInput grenade(Worm &w) = 0;
    virtual IO::PlayerInput cluster(Worm &w) = 0;
    virtual IO::PlayerInput mortar(Worm &w) = 0;
    virtual IO::PlayerInput banana(Worm &w) = 0;
    virtual IO::PlayerInput holy(Worm &w) = 0;
    virtual IO::PlayerInput aerialAttack(Worm &w) = 0;
    virtual IO::PlayerInput dynamite(Worm &w) = 0;
    virtual IO::PlayerInput baseballBat(Worm &w) = 0;
    virtual IO::PlayerInput teleport(Worm &w) = 0;

    virtual IO::PlayerInput startShot(Worm &w) = 0;
    virtual IO::PlayerInput endShot(Worm &w) = 0;
    virtual IO::PlayerInput backFlip(Worm &w) = 0;
    virtual IO::PlayerInput positionSelected(Worm &w) = 0;

    virtual StateID &getState() {
        return this->stateID;
    };

protected:
    StateID stateID;
};

} // namespace Worm

#endif // __WORM_STATE_H__

```

jun 26, 18 2:39	WormStill.cpp	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #include "WormStill.h" #include <iostream> #include "Texture.h" Worm::Still::Still() : State(StateID::Still) {} Worm::Still::~Still() {} void Worm::Still::update(float dt) {} IO::PlayerInput Worm::Still::moveRight(Worm &w) { return IO::PlayerInput::moveRight; } IO::PlayerInput Worm::Still::moveLeft(Worm &w) { return IO::PlayerInput::moveLeft; } IO::PlayerInput Worm::Still::stopMove(Worm &w) { return IO::PlayerInput::stopMove; } IO::PlayerInput Worm::Still::jump(Worm &w) { return IO::PlayerInput::startJump; } IO::PlayerInput Worm::Still::backFlip(Worm &w) { return IO::PlayerInput::startBackFlip; } IO::PlayerInput Worm::Still::bazooka(Worm &w) { return IO::PlayerInput::bazooka; } IO::PlayerInput Worm::Still::pointUp(Worm &w) { return IO::PlayerInput::pointUp; } IO::PlayerInput Worm::Still::pointDown(Worm &w) { return IO::PlayerInput::pointDown; } IO::PlayerInput Worm::Still::startShot(Worm &w) { w.startShot(); return IO::PlayerInput::startShot; } IO::PlayerInput Worm::Still::endShot(Worm &w) { w.endShot(); return IO::PlayerInput::endShot; } IO::PlayerInput Worm::Still::grenade(Worm &w) { return IO::PlayerInput::grenade; } IO::PlayerInput Worm::Still::cluster(Worm &w) { </pre>		

jun 26, 18 2:39	WormStill.cpp	Page 2/2
<pre> return IO::PlayerInput::cluster; } IO::PlayerInput Worm::Still::mortar(Worm &w) { return IO::PlayerInput::mortar; } IO::PlayerInput Worm::Still::banana(Worm &w) { return IO::PlayerInput::banana; } IO::PlayerInput Worm::Still::holy(Worm &w) { return IO::PlayerInput::holy; } IO::PlayerInput Worm::Still::setTimeoutTo(Worm &w, int time) { switch (time) { case 1: return IO::PlayerInput::timeout1; case 2: return IO::PlayerInput::timeout2; case 3: return IO::PlayerInput::timeout3; case 4: return IO::PlayerInput::timeout4; case 5: return IO::PlayerInput::timeout5; default: return IO::PlayerInput::moveNone; } } IO::PlayerInput Worm::Still::aerialAttack(Worm &w) { return IO::PlayerInput::aerialAttack; } IO::PlayerInput Worm::Still::positionSelected(Worm &w) { return IO::PlayerInput::positionSelected; } IO::PlayerInput Worm::Still::dynamite(Worm &w) { return IO::PlayerInput::dynamite; } IO::PlayerInput Worm::Still::teleport(Worm &w) { return IO::PlayerInput::teleport; } IO::PlayerInput Worm::Still::baseballBat(Worm &w) { return IO::PlayerInput::baseballBat; } </pre>		

jun 26, 18 2:39

WormStill.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#ifndef __WORM_QUIET_H__
#define __WORM_QUIET_H__

#include <SDL2/SDL_system.h>

#include "../Worm.h"
#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class Still : public State {
public:
    Still();
    ~Still();
    virtual void update(float dt) override;
    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __WORM_QUIET_H__

```

jun 26, 18 2:39	WormWalk.cpp	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #include "WormWalk.h" #include <iostream> Worm::Walk::Walk() : State(StateID::Walk) {} Worm::Walk::~~Walk() {} void Worm::Walk::update(float dt) {} IO::PlayerInput Worm::Walk::moveLeft(Worm &w) { if (w.direction == Direction::left) { return IO::PlayerInput::moveNone; } return IO::PlayerInput::moveLeft; } IO::PlayerInput Worm::Walk::moveRight(Worm &w) { if (w.direction == Direction::right) { return IO::PlayerInput::moveNone; } return IO::PlayerInput::moveRight; } IO::PlayerInput Worm::Walk::stopMove(Worm &w) { return IO::PlayerInput::stopMove; } IO::PlayerInput Worm::Walk::jump(Worm &w) { return IO::PlayerInput::startJump; } IO::PlayerInput Worm::Walk::backFlip(Worm &w) { return IO::PlayerInput::startBackFlip; } IO::PlayerInput Worm::Walk::bazooka(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::pointUp(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::pointDown(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::startShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::endShot(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::grenade(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39	WormWalk.cpp	Page 2/2
<pre> } IO::PlayerInput Worm::Walk::cluster(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::mortar(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::banana(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::holy(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::setTimeoutTo(Worm &w, int t) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::aerialAttack(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::positionSelected(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::dynamite(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::teleport(Worm &w) { return IO::PlayerInput::moveNone; } IO::PlayerInput Worm::Walk::baseballBat(Worm &w) { return IO::PlayerInput::moveNone; } </pre>		

jun 26, 18 2:39

WormWalk.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#ifndef __WORM_WALK_H__
#define __WORM_WALK_H__

#include <SDL2/SDL_system.h>

#include "../Worm.h"
#include "GameStateMsg.h"
#include "WormState.h"

namespace Worm {
class Walk : public State {
public:
    explicit Walk();
    virtual ~Walk();

    virtual void update(float dt) override;

    virtual IO::PlayerInput moveRight(Worm &w) override;
    virtual IO::PlayerInput moveLeft(Worm &w) override;
    virtual IO::PlayerInput stopMove(Worm &w) override;
    virtual IO::PlayerInput jump(Worm &w) override;
    virtual IO::PlayerInput backFlip(Worm &w) override;
    virtual IO::PlayerInput setTimeoutTo(Worm &w, int t) override;

    virtual IO::PlayerInput bazooka(Worm &w) override;
    virtual IO::PlayerInput grenade(Worm &w) override;
    virtual IO::PlayerInput cluster(Worm &w) override;
    virtual IO::PlayerInput mortar(Worm &w) override;
    virtual IO::PlayerInput banana(Worm &w) override;
    virtual IO::PlayerInput holy(Worm &w) override;
    virtual IO::PlayerInput aerialAttack(Worm &w) override;
    virtual IO::PlayerInput dynamite(Worm &w) override;
    virtual IO::PlayerInput baseballBat(Worm &w) override;
    virtual IO::PlayerInput teleport(Worm &w) override;
    virtual IO::PlayerInput positionSelected(Worm &w) override;

    virtual IO::PlayerInput startShot(Worm &w) override;
    virtual IO::PlayerInput endShot(Worm &w) override;
    virtual IO::PlayerInput pointUp(Worm &w) override;
    virtual IO::PlayerInput pointDown(Worm &w) override;
};
} // namespace Worm

#endif // __WORM_WALK_H__

```


jun 26, 18 2:39

editor.cpp

Page 1/1

```
#include "editor.h"

Editor::Editor::Editor(QWidget *parent) : QGraphicsView(parent) {}

Editor::Editor::~~Editor() {}

void Editor::Editor::wheelEvent(QWheelEvent *event) {
    QGraphicsView::wheelEvent(event);
    if (event->isAccepted()) {
        return;
    }

    static qreal factor = 1.1;

    if (event->angleDelta().y() > 0) {
        scale(factor, factor);
    } else {
        scale(1 / factor, 1 / factor);
    }

    event->accept();
}

void Editor::Editor::mousePressEvent(QMouseEvent *event) {
    QGraphicsView::mousePressEvent(event);
    if (event->isAccepted()) {
        return;
    }

    switch (event->button()) {
        case Qt::LeftButton:
            break;

        case Qt::RightButton:
            break;

        default:
            break;
    }

    event->accept();
}
```

jun 26, 18 2:39

editor.h

Page 1/1

```
#ifndef EDITOR_H
#define EDITOR_H

#include <QGraphicsView>
#include <QWheelEvent>

namespace Editor {
class Editor : public QGraphicsView {
public:
    Editor(QWidget *parent);
    ~Editor();

    void wheelEvent(QWheelEvent *event);
    void mousePressEvent(QMouseEvent *event);
};
}

// Q_DECLARE_METATYPE(Editor::Editor);

#endif // EDITOR_H
```

jun 26, 18 2:39

Editor.pro

Page 1/3

```
#-----  
#  
# Project created by QtCreator 2018-06-02T20:55:03  
#  
#-----  
QT      += core gui  
QMAKE_CXXFLAGS += -std=c++0x  
  
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  
  
TARGET = Editor  
TEMPLATE = app  
  
SOURCES += main.cpp\  
          mainwindow.cpp \  
          editor.cpp \  
          editorview.cpp \  
          editorscene.cpp \  
          stagedata.cpp \  
          stageelement.cpp \  
          stageelementworm.cpp \  
          stageelemllonggirder.cpp \  
          stageelemshortgirder.cpp \  
          qgraphicsitemlayer.cpp \  
          ../../libs/yaml-cpp/contrib/graphbuilder.cpp \  
          ../../libs/yaml-cpp/contrib/graphbuilderadapter.cpp \  
          ../../libs/yaml-cpp/binary.cpp \  
          ../../libs/yaml-cpp/convert.cpp \  
          ../../libs/yaml-cpp/directives.cpp \  
          ../../libs/yaml-cpp/emit.cpp \  
          ../../libs/yaml-cpp/emitfromevents.cpp \  
          ../../libs/yaml-cpp/emitter.cpp \  
          ../../libs/yaml-cpp/emitterstate.cpp \  
          ../../libs/yaml-cpp/emitterutils.cpp \  
          ../../libs/yaml-cpp/exceptions.cpp \  
          ../../libs/yaml-cpp/exp.cpp \  
          ../../libs/yaml-cpp/memory.cpp \  
          ../../libs/yaml-cpp/node.cpp \  
          ../../libs/yaml-cpp/node_data.cpp \  
          ../../libs/yaml-cpp/nodebuilder.cpp \  
          ../../libs/yaml-cpp/nodeevents.cpp \  
          ../../libs/yaml-cpp/null.cpp \  
          ../../libs/yaml-cpp/ostream_wrapper.cpp \  
          ../../libs/yaml-cpp/parse.cpp \  
          ../../libs/yaml-cpp/parser.cpp \  
          ../../libs/yaml-cpp/regex_yaml.cpp \  
          ../../libs/yaml-cpp/scanner.cpp \  
          ../../libs/yaml-cpp/scanscalar.cpp \  
          ../../libs/yaml-cpp/scantag.cpp \  
          ../../libs/yaml-cpp/scantoken.cpp \  
          ../../libs/yaml-cpp/simplekey.cpp \  
          ../../libs/yaml-cpp/singledocparser.cpp \  
          ../../libs/yaml-cpp/stream.cpp \  
          ../../libs/yaml-cpp/tag.cpp  
  
HEADERS += mainwindow.h \  
          editor.h \  
          editorview.h \  
          editorscene.h \  
          stagedata.h \
```

jun 26, 18 2:39

Editor.pro

Page 2/3

```
stageelement.h \  
stageelementworm.h \  
stageelemllonggirder.h \  
stageelemshortgirder.h \  
qgraphicsitemlayer.h \  
../../libs/yaml-cpp/contrib/anchordict.h \  
../../libs/yaml-cpp/contrib/graphbuilder.h \  
../../libs/yaml-cpp/contrib/graphbuilderadapter.h \  
../../libs/yaml-cpp/node/detail/bool_type.h \  
../../libs/yaml-cpp/node/detail/impl.h \  
../../libs/yaml-cpp/node/detail/iterator.h \  
../../libs/yaml-cpp/node/detail/iterator_fwd.h \  
../../libs/yaml-cpp/node/detail/memory.h \  
../../libs/yaml-cpp/node/detail/node.h \  
../../libs/yaml-cpp/node/detail/node_data.h \  
../../libs/yaml-cpp/node/detail/node_iterator.h \  
../../libs/yaml-cpp/node/detail/node_ref.h \  
../../libs/yaml-cpp/node/convert.h \  
../../libs/yaml-cpp/node/emit.h \  
../../libs/yaml-cpp/node/impl.h \  
../../libs/yaml-cpp/node/iterator.h \  
../../libs/yaml-cpp/node/node.h \  
../../libs/yaml-cpp/node/parse.h \  
../../libs/yaml-cpp/node/ptr.h \  
../../libs/yaml-cpp/node/type.h \  
../../libs/yaml-cpp/anchor.h \  
../../libs/yaml-cpp/binary.h \  
../../libs/yaml-cpp/collectionstack.h \  
../../libs/yaml-cpp/directives.h \  
../../libs/yaml-cpp/dll.h \  
../../libs/yaml-cpp/emitfromevents.h \  
../../libs/yaml-cpp/emitter.h \  
../../libs/yaml-cpp/emitterdef.h \  
../../libs/yaml-cpp/emittermanip.h \  
../../libs/yaml-cpp/emitterstate.h \  
../../libs/yaml-cpp/emitterstyle.h \  
../../libs/yaml-cpp/emitterutils.h \  
../../libs/yaml-cpp/eventhandler.h \  
../../libs/yaml-cpp/exceptions.h \  
../../libs/yaml-cpp/exp.h \  
../../libs/yaml-cpp/indentation.h \  
../../libs/yaml-cpp/mark.h \  
../../libs/yaml-cpp/nodebuilder.h \  
../../libs/yaml-cpp/nodeevents.h \  
../../libs/yaml-cpp/noncopyable.h \  
../../libs/yaml-cpp/null.h \  
../../libs/yaml-cpp/ostream_wrapper.h \  
../../libs/yaml-cpp/parser.h \  
../../libs/yaml-cpp/ptr_vector.h \  
../../libs/yaml-cpp/regex_yaml.h \  
../../libs/yaml-cpp/regeximpl.h \  
../../libs/yaml-cpp/scanner.h \  
../../libs/yaml-cpp/scanscalar.h \  
../../libs/yaml-cpp/scantag.h \  
../../libs/yaml-cpp/setting.h \  
../../libs/yaml-cpp/singledocparser.h \  
../../libs/yaml-cpp/stlemmitter.h \  
../../libs/yaml-cpp/stream.h \  
../../libs/yaml-cpp/streamcharsource.h \  
../../libs/yaml-cpp/stringsource.h \  
../../libs/yaml-cpp/tag.h \  
../../libs/yaml-cpp/token.h \
```

jun 26, 18 2:39	Editor.pro	Page 3/3
<pre>../../../../libs/yaml-cpp/traits.h \ ../../../../libs/yaml-cpp/yaml.h FORMS += mainwindow.ui RESOURCES += \ resources.qrc INCLUDEPATH += ../../../../libs</pre>		

jun 26, 18 7:40

editorscene.cpp

Page 1/3

```

#include "editorscene.h"
#include <QDebug>
#include <QGraphicsPixmapItem>
#include <QImage>
#include <QMouseEvent>
#include <QPainter>

EditorScene::EditorScene(QRectF rect) : QGraphicsScene(nullptr), rect(rect) {
    this->setSceneRect(rect);
}

void EditorScene::setCursor(StageElement *newCursor) {
    if (this->cursor) {
        delete this->cursor;
    }
    this->cursor = newCursor;
    QGraphicsScene::addItem(this->cursor);
}

void EditorScene::hideCursor() {
    if (this->cursor) {
        QGraphicsScene::removeItem(this->cursor);
    }
}

void EditorScene::showCursor() {
    if (this->cursor) {
        QGraphicsScene::addItem(this->cursor);
    }
}

void EditorScene::addItem(QGraphicsItem *elem) {
    if (elem->scene() != this) {
        QGraphicsScene::addItem(elem);
    }
}

void EditorScene::addItem(StageElement *elem) {
    if (elem->scene() != this) {
        if (!this->rect.contains(elem->getPosition())) {
            return;
        }
        QGraphicsScene::addItem(elem);
        this->elements.insert(elem);
    }
}

void EditorScene::removeItem(StageElement *elem) {
    if (this->contains(elem)) {
        this->elements.erase(this->elements.find(elem));

        if (elem->scene()) {
            QGraphicsScene::removeItem(elem);
        }
    }
}

void EditorScene::removeItem(QGraphicsItem *elem) {
    if (elem->scene()) {
        QGraphicsScene::removeItem(elem);
    }
}

```

jun 26, 18 7:40

editorscene.cpp

Page 2/3

```

void EditorScene::serialize(StageData &sd) {
    sd.bgColor = this->bgColor;
    for (auto *elem : this->elements) {
        elem->serialize(sd);
    }
}

QList<StageElement *> EditorScene::collidingItems(StageElement *elem) {
    QList<StageElement *> rv;
    for (QGraphicsItem *other : QGraphicsScene::collidingItems(elem)) {
        if (other == elem) {
            continue;
        }

        if (this->contains(dynamic_cast<StageElement *>(other))) {
            rv.append(dynamic_cast<StageElement *>(other));
        }
    }
    return rv;
}

bool EditorScene::contains(StageElement *elem) {
    auto it = this->elements.find(elem);
    return (it != this->elements.end());
}

void EditorScene::setBgColor(QColor color) {
    this->bgColor = color;
    if (this->bgColorLayer) {
        this->removeItem(this->bgColorLayer);
        delete this->bgColorLayer;
    }

    this->bgColorLayer = new QGraphicsItemLayer;
    this->bgColorLayer->setZValue(-4);
    this->addItem(this->bgColorLayer);
    QGraphicsRectItem *bg = new QGraphicsRectItem(this->rect, this->bgColorLayer);
    bg->setBrush(QBrush{color});
}

void EditorScene::setFartherBg(QImage image) {
    this->setBackground(image, &this->fartherBg, -3);
}

void EditorScene::setMedianBg(QImage image) {
    this->setBackground(image, &this->medianBg, -2);
}

void EditorScene::setCloserBg(QImage image) {
    this->setBackground(image, &this->closeBg, -1);
}

void EditorScene::setBackground(QImage image, QGraphicsItemLayer **layerPtr, qreal zValue) {
    if (*layerPtr) {
        this->removeItem(*layerPtr);
        delete *layerPtr;
    }

    *layerPtr = new QGraphicsItemLayer;
}

```

jun 26, 18 7:40

editorscene.cpp

Page 3/3

```
QGraphicsItemLayer *layer = *layerPtr;

layer->setZValue(zValue);
this->addItem(layer);

for (int i = 0; i < this->rect.width() / image.width() + 1; i++) {
    QGraphicsPixmapItem *pix = new QGraphicsPixmapItem(layer);
    pix->setPixmap(QPixmap::fromImage(image));
    pix->setPos(image.width() * i, this->rect.height() - image.height());
}
}
```

jun 26, 18 7:40

editorscene.h

Page 1/1

```

#ifndef EDITORSCENE_H
#define EDITORSCENE_H

#include <QColor>
#include <QGraphicsScene>
#include <QImage>
#include <QObject>
#include <QWidget>
#include <set>
#include <string>
#include "qgraphicsitemlayer.h"
#include "stageelement.h"

class EditorScene : public QGraphicsScene {
    Q_OBJECT

public:
    EditorScene(QRectF rect);

    void setCursor(StageElement *newCursor);
    void hideCursor();
    void showCursor();

    void addItem(QGraphicsItem *elem);
    void addItem(StageElement *elem);
    void removeItem(QGraphicsItem *elem);
    void removeItem(StageElement *elem);

    virtual QList<StageElement *> collidingItems(StageElement *elem);
    bool contains(StageElement *elem);

    void serialize(StageData &sd);

    /* background */
    void setBgColor(QColor color);
    void setFartherBg(QImage image);
    void setMedianBg(QImage image);
    void setCloserBg(QImage image);

private:
    void setBackground(QImage image, QGraphicsItemLayer **layerPtr, qreal zValue
);

    QRectF rect;
    QColor bgColor{Qt::white};
    QGraphicsItemLayer *closeBg{nullptr};
    QGraphicsItemLayer *medianBg{nullptr};
    QGraphicsItemLayer *fartherBg{nullptr};
    QGraphicsItemLayer *bgColorLayer{nullptr};
    StageElement *cursor{nullptr};
    std::string resource;
    std::set<StageElement *> elements;
};

#endif // EDITORSCENE_H

```

jun 26, 18 7:40

editorview.cpp

Page 1/3

```

#include "editorview.h"
#include <QGraphicsPixmapItem>
#include <QImage>
#include <QtDebug>
#include <QScrollBar>
#include <cmath>
#include "stageelementworm.h"
#include "stageelemlonggirder.h"
#include "stageelemshortgirder.h"

const qreal cursorOpacity = 0.7;

EditorView::EditorView(QWidget *parent) : QGraphicsView(parent) {
    this->setAttribute(Qt::WA_Hover, true);
    this->setTransformationAnchor(QGraphicsView::AnchorUnderMouse);
    this->setLongGirder();
}

void EditorView::drawCloseBg(QString &) {}

void EditorView::setScene(EditorScene *scene) {
    QGraphicsView::setScene(scene);
    this->escene = scene;
    this->horizontalScrollBar()->setValue(this->horizontalScrollBar()->maximum()
/ 2);
    this->verticalScrollBar()->setValue(this->verticalScrollBar()->maximum());
}

void EditorView::setWorm() {
    if (this->stageElem) {
        delete this->stageElem;
    }

    this->stageElem = new StageElementWorm{cursorOpacity};
}

void EditorView::setShortGirder() {
    if (this->stageElem) {
        delete this->stageElem;
    }

    this->stageElem = new StageElemShortGirder{cursorOpacity};
}

void EditorView::setLongGirder() {
    if (this->stageElem) {
        delete this->stageElem;
    }

    this->stageElem = new StageElemLongGirder{cursorOpacity};
}

void EditorView::mousePressEvent(QMouseEvent *) {}

void EditorView::deleteAt(QPoint pos) {
    this->scene()->removeItem(this->stageElem);
    QGraphicsItem *item = this->itemAt(pos);
    if (item) {
        this->escene->removeItem(static_cast<StageElement *>(item));
    }
    this->escene->addItem(this->stageElem);
}

```

jun 26, 18 7:40

editorview.cpp

Page 2/3

```

void EditorView::keyPressEvent(QKeyEvent *event) {
    if (event->key() == Qt::Key_Plus) {
        this->stageElem->increaseAngle();
    } else if (event->key() == Qt::Key_Minus) {
        this->stageElem->decreaseAngle();
    }
}

void EditorView::createAt(QPoint pos) {
    if (!this->stageElem) {
        return;
    }

    if (this->collides()) {
        return;
    }

    StageElement *newElem = this->stageElem->clone();

    QPointF lpos = this->mapToScene(pos);
    lpos.rx() -= newElem->pixmap().width() / 2;
    lpos.ry() -= newElem->pixmap().height() / 2;

    newElem->setPos(lpos);
    this->escene->addItem(newElem);
}

void EditorView::mouseReleaseEvent(QMouseEvent *event) {
    event->accept();

    if (event->button() & Qt::RightButton) {
        this->deleteAt(event->pos());
    } else {
        this->createAt(event->pos());
        this->stageElem->setZValue(1);
    }
}

void EditorView::mouseMoveEvent(QMouseEvent *event) {
    if (!this->stageElem) {
        return;
    }

    /* set the position of the hint image under the mouse */
    QPointF pos = this->mapToScene(event->pos());
    pos.rx() -= this->stageElem->pixmap().width() / 2;
    pos.ry() -= this->stageElem->pixmap().height() / 2;

    this->stageElem->setPos(pos);
    event->accept();
}

bool EditorView::event(QEvent *event) {
    switch (event->type()) {
        case QEvent::HoverEnter:
            if (this->stageElem) {
                this->setFocus();
                this->escene->addItem(dynamic_cast<QGraphicsItem *>(this->stageE
lem));
            }
            return true;
    }
}

```


jun 26, 18 7:40

editorview.cpp

Page 3/3

```

        case QEvent::HoverLeave:
            if (this->stageElem) {
                this->escene->removeItem(dynamic_cast<QGraphicsItem *>(this->stageElem));
            }
            return true;
        default:
            break;
    }

    return QGraphicsView::event(event);
}

void EditorView::wheelEvent(QWheelEvent *event) {
    static qreal factor = 1.1;

    if (event->delta() > 0) {
        this->scale(factor, factor);
    } else {
        this->scale(1.0 / factor, 1.0 / factor);
    }

    /* set the position of the hint image under the mouse */
    QPointF pos = this->mapToScene(event->pos());
    pos.rx() -= this->stageElem->pixmap().width() / 2;
    pos.ry() -= this->stageElem->pixmap().height() / 2;

    this->stageElem->setPos(pos);
    event->accept();
}

bool EditorView::collides() {
    for (StageElement *other : this->escene->collidingItems(this->stageElem)) {
        if (!this->stageElem->canOverlap(other)) {
            return true;
        }
    }

    return false;
}

void EditorView::serialize(StageData &sd) const {
    if (this->stageElem) {
        this->escene->removeItem(this->stageElem);
    }

    this->escene->serialize(sd);

    if (this->stageElem) {
        this->escene->addItem(this->stageElem);
    }
}

```

jun 26, 18 2:39

editorview.h

Page 1/1

```

#ifndef EDITORVIEW_H
#define EDITORVIEW_H

#include <QEvent>
#include <QGraphicsView>
#include <QObject>
#include <QWheelEvent>
#include <QWidget>
#include "editorscene.h"
#include "stagedata.h"
#include "stageelement.h"

class EditorView : public QGraphicsView {
    Q_OBJECT

public:
    EditorView(QWidget *parent);
    virtual void setScene(EditorScene *scene);

    void drawCloseBg(QString &fileName);

public slots:
    void setWorm();
    void setShortGirder();
    void setLongGirder();

    void serialize(StageData &sd) const;

    // QWidget interface
protected:
    void mousePressEvent(QMouseEvent *event);
    void mouseReleaseEvent(QMouseEvent *);
    void mouseMoveEvent(QMouseEvent *);
    void wheelEvent(QWheelEvent *);
    bool event(QEvent *event);
    void hoverEvent(QHoverEvent *event);
    void keyPressEvent(QKeyEvent *event);

    bool collides();
    void deleteAt(QPoint pos);
    void createAt(QPoint pos);

private:
    StageElement *stageElem{nullptr};
    EditorScene *escene{nullptr};
};

#endif // EDITORVIEW_H

```

jun 26, 18 2:39

main.cpp

Page 1/1

```
#include <QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[]) {
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

jun 29, 18 16:28	mainwindow.cpp	Page 1/3
<pre> #include "mainwindow.h" #include <QColor> #include <QColorDialog> #include <QErrorMessage> #include <QFileDialog> #include <fstream> #include "stagedata.h" #include "ui_mainwindow.h" MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWi ndow) { ui->setupUi(this); this->scene = new EditorScene(this->stageSize); this->ui->editorView->setScene(this->scene); this->ui->colorPreview->setScene(new QGraphicsScene); /* toolbar */ connect(this->ui->actionAdd_Worm, SIGNAL(triggered(bool)), this->ui->editorV iew, SLOT(setWorm())); connect(this->ui->actionAdd_Long_Girder, SIGNAL(triggered(bool)), this->ui-> editorView, SLOT(setLongGirder())); connect(this->ui->actionShort_Girder, SIGNAL(triggered(bool)), this->ui->edi torView, SLOT(setShortGirder())); QColor defaultColor{0xba, 0x8d, 0xc6}; this->scene->setBgColor(defaultColor); this->ui->colorPreview->setBackgroundBrush(QBrush(defaultColor)); this->showMaximized(); } MainWindow::~MainWindow() { delete ui; delete scene; } void MainWindow::on_actionLejano_triggered() { QString fileName = QFileDialog::getOpenFileName(this, tr("Seleccione una imagen para el fondo lejano"), "/home", tr("Image Files (*.png)")); if (!fileName.isEmpty()) { this->fartherBgFile = fileName; this->scene->setFartherBg(QImage(fileName)); } } void MainWindow::on_actionMedio_triggered() { QString fileName = QFileDialog::getOpenFileName(this, tr("Seleccione una imagen para el fondo medio"), "/home", tr("Image Files (*.png)")); if (!fileName.isEmpty()) { this->midBgFile = fileName; this->scene->setMedianBg(QImage(fileName)); } } void MainWindow::on_actionCercano_triggered() { </pre>		

jun 29, 18 16:28	mainwindow.cpp	Page 2/3
<pre> QString fileName = QFileDialog::getOpenFileName(this, tr("Seleccione una imagen para el fondo cercano") , "/home", tr("Image Files (*.png)")); if (!fileName.isEmpty()) { this->closeBgFile = fileName; this->scene->setCloserBg(QImage(fileName)); } } void MainWindow::on_bgColorButton_clicked() { QColor color = QColorDialog::getColor(Qt::white, this); if (color.isValid()) { this->scene->setBgColor(color); this->ui->colorPreview->setBackgroundBrush(QBrush(color)); } } void MainWindow::on_actionOpen_triggered() { /* serializes the stage */ StageData sd{this->stageSize.width(), stageSize.height()}; sd.closeBgFile = this->closeBgFile; sd.medianBgFile = this->midBgFile; sd.fartherBgFile = this->fartherBgFile; sd.wormsHealth = this->ui->wormsHP->value(); sd.numPlayers= this->ui->numPlayers->value(); /* weapon ammo */ const QString WEAPON_PREFIX = "wpn_"; for(auto *child : this->ui->stageParams->children()) { if(child->objectName().startsWith(WEAPON_PREFIX)) { QString weaponName = child->objectName().remove(0, WEAPON_PREFIX.siz e()); QSpinBox *widget = dynamic_cast<QSpinBox *>(child); sd.addWeaponAmmo(weaponName, widget->value()); } } this->ui->editorView->serialize(sd); if (static_cast<int>(sd.numWorms()) < sd.numPlayers) { QErrorMessage::qtHandler()->showMessage("Se necesita al menos 1 worm por jugador"); return; } /* gets the output file name */ QString fileName = QFileDialog::getSaveFileName(this, tr("Nombre de archivo de sali da"), "/home", tr("YAML (*.yaml)")); /* checks if a file was selected */ if (fileName.isEmpty()) { return; } std::ofstream file; file.open(fileName.toStdString(), std::ios::out std::ios::trunc); if (!file) { QErrorMessage::qtHandler()->showMessage("Error al abrir el archivo"); return; } </pre>		

jun 29, 18 16:28

mainwindow.cpp

Page 3/3

```
}
/* gets the base name of the file */
QStringList list = fileName.split('/');
QList<QString>::Iterator it = list.end();
it--;
QStringList list2 = it->split('.');
sd.dump(file, list2[0].toString());
}
```

jun 26, 18 7:40

mainwindow.h

Page 1/1

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QGraphicsScene>
#include <QGraphicsView>
#include <QMainWindow>
#include <QString>
#include "editorscene.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow {
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_actionLejano_triggered();

    void on_actionMedio_triggered();

    void on_actionCercano_triggered();

    void on_bgColorButton_clicked();

    void on_actionOpen_triggered();

private:
    Ui::MainWindow *ui;
    QRectF stageSize{0, 0, 13 * 250, 13 * 250};
    EditorScene *scene;
    QString closeBgFile;
    QString midBgFile;
    QString fartherBgFile;
};

#endif // MAINWINDOW_H
```

jun 26, 18 7:40	mainwindow.ui	Page 1/11
<pre> <?xml version="1.0" encoding="UTF-8"?> <ui version="4.0"> <class>MainWindow</class> <widget class="QMainWindow" name="MainWindow"> <property name="geometry"> <rect> <x>0</x> <y>0</y> <width>1065</width> <height>609</height> </rect> </property> <property name="windowTitle"> <string>MainWindow</string> </property> <widget class="QWidget" name="centralWidget"> <property name="sizePolicy"> <sizepolicy hsizepolicy="Maximum" vsizetype="Maximum"> <horstretch>0</horstretch> <verstretch>0</verstretch> </sizepolicy> </property> <layout class="QGridLayout" name="gridLayout"> <item row="0" column="1"> <widget class="QScrollArea" name="stageParamsContainer"> <property name="minimumSize"> <size> <width>250</width> <height>200</height> </size> </property> <property name="maximumSize"> <size> <width>250</width> <height>16777215</height> </size> </property> <property name="widgetResizable"> <bool>true</bool> </property> <widget class="QWidget" name="stageParams"> <property name="geometry"> <rect> <x>0</x> <y>0</y> <width>235</width> <height>617</height> </rect> </property> <layout class="QFormLayout" name="formLayout"> <item row="0" column="0" colspan="2"> <widget class="QLabel" name="label_2"> <property name="frameShadow"> <enum>QFrame::Plain</enum> </property> <property name="lineWidth"> <number>1</number> </property> <property name="text"> <string>&lt;b>Worms</string> </property> <property name="textFormat"> </pre>		

jun 26, 18 7:40	mainwindow.ui	Page 2/11
<pre> <enum>Qt::RichText</enum> </property> <property name="scaledContents"> <bool>false</bool> </property> <property name="alignment"> <set>Qt::AlignCenter</set> </property> <property name="wordWrap"> <bool>false</bool> </property> <property name="textInteractionFlags"> <set>Qt::NoTextInteraction</set> </property> </widget> </item> <item row="2" column="0"> <widget class="QLabel" name="label"> <property name="text"> <string>HP</string> </property> </widget> </item> <item row="2" column="1"> <widget class="QSpinBox" name="wormsHP"> <property name="maximum"> <number>250</number> </property> <property name="singleStep"> <number>10</number> </property> <property name="value"> <number>100</number> </property> </widget> </item> <item row="4" column="0" colspan="2"> <widget class="Line" name="line"> <property name="orientation"> <enum>Qt::Horizontal</enum> </property> </widget> </item> <item row="5" column="0"> <spacer name="verticalSpacer"> <property name="orientation"> <enum>Qt::Vertical</enum> </property> <property name="sizeHint" stdset="0"> <size> <width>20</width> <height>10</height> </size> </property> </spacer> </item> <item row="6" column="0" colspan="2"> <widget class="QLabel" name="label_3"> <property name="text"> <string>&lt;html>&lt;head>&lt;body>&lt;p>&lt;span style= "font-weight: 600;">Turno&lt;/span>&lt;/p>&lt;/body>&lt;/h tml>&lt;/string> </pre>		

jun 26, 18 7:40	mainwindow.ui	Page 3/11
	<pre> </property> <property name="alignment"> <set>Qt::AlignCenter</set> </property> </widget> </item> <item row="7" column="0"> <widget class="QLabel" name="label_4"> <property name="text"> <string>Duraci3n [s]</string> </property> </widget> </item> <item row="7" column="1"> <widget class="QSpinBox" name="spinBox_2"> <property name="maximum"> <number>120</number> </property> <property name="singleStep"> <number>5</number> </property> <property name="value"> <number>40</number> </property> </widget> </item> <item row="8" column="0" colspan="2"> <widget class="Line" name="line_2"> <property name="minimumSize"> <size> <width>50</width> <height>0</height> </size> </property> <property name="orientation"> <enum>Qt::Horizontal</enum> </property> </widget> </item> <item row="9" column="0"> <spacer name="verticalSpacer_2"> <property name="orientation"> <enum>Qt::Vertical</enum> </property> <property name="sizeHint" stdset="0"> <size> <width>20</width> <height>10</height> </size> </property> </spacer> </item> <item row="10" column="0" colspan="2"> <widget class="QLabel" name="label_5"> <property name="text"> <string>Armas</string> </property> <property name="alignment"> <set>Qt::AlignCenter</set> </property> </widget> </item> </pre>	

jun 26, 18 7:40	mainwindow.ui	Page 4/11
	<pre> <item row="11" column="0"> <widget class="QLabel" name="label_6"> <property name="text"> <string>Bazooka</string> </property> </widget> </item> <item row="11" column="1"> <widget class="QSpinBox" name="wpn_bazooka"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>999</number> </property> </widget> </item> <item row="12" column="0"> <widget class="QLabel" name="label_7"> <property name="text"> <string>Granada Verde</string> </property> </widget> </item> <item row="12" column="1"> <widget class="QSpinBox" name="wpn_grenade"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>999</number> </property> </widget> </item> <item row="13" column="0"> <widget class="QLabel" name="label_8"> <property name="text"> <string>Granada Roja</string> </property> </widget> </item> <item row="13" column="1"> <widget class="QSpinBox" name="wpn_cluster"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>5</number> </property> </widget> </item> <item row="14" column="0"> <widget class="QLabel" name="label_9"> <property name="text"> <string>Mortero</string> </property> </widget> </item> <item row="14" column="1"> <widget class="QSpinBox" name="wpn_mortar"> <property name="value"> <number>5</number> </property> </widget> </item> </pre>	

jun 26, 18 7:40	mainwindow.ui	Page 5/11
	<pre> </property> </widget> </item> <item row="15" column="0"> <widget class="QLabel" name="label_10"> <property name="text"> <string>Banana</string> </property> </widget> </item> <item row="15" column="1"> <widget class="QSpinBox" name="wpn_banana"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>5</number> </property> </widget> </item> <item row="16" column="0"> <widget class="QLabel" name="label_11"> <property name="text"> <string>Granada Santa</string> </property> </widget> </item> <item row="16" column="1"> <widget class="QSpinBox" name="wpn_holy"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>2</number> </property> </widget> </item> <item row="17" column="0"> <widget class="QLabel" name="label_12"> <property name="text"> <string>Dinamita</string> </property> </widget> </item> <item row="17" column="1"> <widget class="QSpinBox" name="wpn_dynamite"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>2</number> </property> </widget> </item> <item row="18" column="0"> <widget class="QLabel" name="label_13"> <property name="text"> <string>Aereo</string> </property> </widget> </item> <item row="18" column="1"> </pre>	

jun 26, 18 7:40	mainwindow.ui	Page 6/11
	<pre> <widget class="QSpinBox" name="wpn_aerialAttack"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>1</number> </property> </widget> </item> <item row="19" column="0"> <widget class="QLabel" name="label_14"> <property name="text"> <string>Bate </string> </property> </widget> </item> <item row="19" column="1"> <widget class="QSpinBox" name="wpn_baseballBat"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>1</number> </property> </widget> </item> <item row="20" column="0"> <widget class="QLabel" name="label_15"> <property name="text"> <string>TeletransportaciÃ³n</string> </property> </widget> </item> <item row="20" column="1"> <widget class="QSpinBox" name="wpn_teleport"> <property name="maximum"> <number>999</number> </property> <property name="value"> <number>2</number> </property> </widget> </item> <item row="21" column="0" colspan="2"> <widget class="Line" name="line_3"> <property name="minimumSize"> <size> <width>100</width> <height>0</height> </size> </property> <property name="orientation"> <enum>Qt::Horizontal</enum> </property> </widget> </item> <item row="22" column="0"> <spacer name="verticalSpacer_4"> <property name="orientation"> <enum>Qt::Vertical</enum> </property> <property name="sizeHint" stdset="0"> </pre>	

jun 26, 18 7:40

mainwindow.ui

Page 7/11

```

        <size>
        <width>20</width>
        <height>10</height>
        </size>
    </property>
</spacer>
</item>
<item row="23" column="0" colspan="2">
    <widget class="QLabel" name="label_16">
        <property name="text">
            <string>&lt;b>Fondo</b></string>
        </property>
        <property name="alignment">
            <set>Qt::AlignCenter</set>
        </property>
    </widget>
</item>
<item row="24" column="0">
    <widget class="QPushButton" name="bgColorButton">
        <property name="text">
            <string>Color</string>
        </property>
    </widget>
</item>
<item row="24" column="1">
    <widget class="QGraphicsView" name="colorPreview">
        <property name="enabled">
            <bool>true</bool>
        </property>
        <property name="sizePolicy">
            <sizepolicy hsize="Ignored" vsize="Ignored">
                <horstretch>0</horstretch>
                <verstretch>0</verstretch>
            </sizepolicy>
        </property>
        <property name="backgroundBrush">
            <brush brushstyle="SolidPattern">
                <color alpha="255">
                    <red>255</red>
                    <green>255</green>
                    <blue>255</blue>
                </color>
            </brush>
        </property>
        <property name="foregroundBrush">
            <brush brushstyle="NoBrush">
                <color alpha="255">
                    <red>186</red>
                    <green>141</green>
                    <blue>198</blue>
                </color>
            </brush>
        </property>
    </widget>
</item>
<item row="3" column="1">
    <widget class="QSpinBox" name="numPlayers">
        <property name="minimum">
            <number>1</number>
        </property>
        <property name="maximum">
            <number>4</number>

```

jun 26, 18 7:40

mainwindow.ui

Page 8/11

```

    </property>
</widget>
</item>
<item row="3" column="0">
    <widget class="QLabel" name="label_17">
        <property name="text">
            <string>Jugadores</string>
        </property>
    </widget>
</item>
</layout>
</widget>
</widget>
</item>
<item row="0" column="0">
    <widget class="EditorView" name="editorView">
        <property name="sizePolicy">
            <sizepolicy hsize="Expanding" vsize="Expanding">
                <horstretch>0</horstretch>
                <verstretch>0</verstretch>
            </sizepolicy>
        </property>
        <property name="minimumSize">
            <size>
                <width>500</width>
                <height>500</height>
            </size>
        </property>
        <property name="cursor" stdset="0">
            <cursorShape>BlankCursor</cursorShape>
        </property>
        <property name="layoutDirection">
            <enum>Qt::RightToLeft</enum>
        </property>
        <property name="sizeAdjustPolicy">
            <enum>QAbstractScrollArea::AdjustToContents</enum>
        </property>
        <property name="alignment">
            <set>Qt::AlignCenter</set>
        </property>
        <property name="renderHints">
            <set>QPainter::SmoothPixmapTransform|QPainter::TextAntialiasing</set>
        </property>
        <property name="dragMode">
            <enum>QGraphicsView::NoDrag</enum>
        </property>
        <property name="transformationAnchor">
            <enum>QGraphicsView::AnchorUnderMouse</enum>
        </property>
        <property name="resizeAnchor">
            <enum>QGraphicsView::AnchorUnderMouse</enum>
        </property>
        <property name="viewportUpdateMode">
            <enum>QGraphicsView::BoundingRectViewportUpdate</enum>
        </property>
        <property name="rubberBandSelectionMode">
            <enum>Qt::IntersectsItemBoundingRect</enum>
        </property>
    </widget>
</item>
</layout>
</widget>

```

jun 26, 18 7:40	mainwindow.ui	Page 9/11
<pre> <widget class="QMenuBar" name="menuBar"> <property name="geometry"> <rect> <x>0</x> <y>0</y> <width>1065</width> <height>25</height> </rect> </property> <widget class="QMenu" name="menuEditor"> <property name="title"> <string>File</string> </property> <addaction name="actionOpen"/> </widget> <widget class="QMenu" name="menuFondo"> <property name="title"> <string>Fondo</string> </property> <addaction name="actionLejano"/> <addaction name="actionMedio"/> <addaction name="actionCercano"/> </widget> <addaction name="menuEditor"/> <addaction name="menuFondo"/> </widget> <widget class="QToolBar" name="mainToolBar"> <property name="sizePolicy"> <sizepolicy hsize="Preferred" vsize="Fixed"> <horstretch>0</horstretch> <verstretch>0</verstretch> </sizepolicy> </property> <property name="iconSize"> <size> <width>50</width> <height>50</height> </size> </property> <attribute name="toolBarArea"> <enum>TopToolBarArea</enum> </attribute> <attribute name="toolBarBreak"> <bool>false</bool> </attribute> <addaction name="actionAdd_Worm"/> <addaction name="actionAdd_Long_Girder"/> <addaction name="actionShort_Girder"/> </widget> <action name="actionOpen"> <property name="text"> <string>Export</string> </property> </action> <action name="actionAdd_Worm"> <property name="icon"> <iconset resource="resources.qrc"> <normaloff>:/assets/buttons/worm.png</normaloff>:/assets/buttons/worm.png</iconset> </property> <property name="text"> <string>Add Worm</string> </pre>		

jun 26, 18 7:40	mainwindow.ui	Page 10/11
<pre> </property> <property name="toolTip"> <string>&lt;html&gt;&lt;head&gt;&lt;body&gt;&lt;p&gt;Add Worm&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string> </property> <property name="shortcut"> <string>W</string> </property> </action> <action name="actionAdd_Long_Girder"> <property name="icon"> <iconset resource="resources.qrc"> <normaloff>:/assets/buttons/long-girder.png</normaloff>:/assets/buttons/long-girder.png</iconset> </property> <property name="text"> <string>Long Girder</string> </property> <property name="toolTip"> <string>&lt;html&gt;&lt;head&gt;&lt;body&gt;&lt;p&gt;Add long girder&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string> </property> <property name="shortcut"> <string>L</string> </property> </action> <action name="actionShort_Girder"> <property name="icon"> <iconset resource="resources.qrc"> <normaloff>:/assets/buttons/short-girder.png</normaloff>:/assets/buttons/short-girder.png</iconset> </property> <property name="text"> <string>Short Girder</string> </property> <property name="toolTip"> <string>&lt;html&gt;&lt;head&gt;&lt;body&gt;&lt;p&gt;Add short girder&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string> </property> <property name="shortcut"> <string>S</string> </property> </action> <action name="actionLejano"> <property name="text"> <string>Lejano</string> </property> </action> <action name="actionMedio"> <property name="text"> <string>Medio</string> </property> </action> <action name="actionCercano"> <property name="text"> <string>Cercano</string> </property> </action> </widget> <layoutdefault spacing="6" margin="11"/> <customwidgets> <customwidget> </pre>		

jun 26, 18 7:40

mainwindow.ui

Page 11/11

```
<class>EditorView</class>
<extends>QGraphicsView</extends>
<header>editorview.h</header>
<slots>
    <slot>setWorm()</slot>
</slots>
</customwidget>
</customwidgets>
<resources>
    <include location="resources.qrc"/>
</resources>
<connections/>
</ui>
```

jun 26, 18 2:39

qgraphicsitemlayer.cpp

Page 1/1

```
#include "qgraphicsitemlayer.h"
```

```
QGraphicsItemLayer::QGraphicsItemLayer() : QGraphicsItem(nullptr) {}
```

```
QRectF QGraphicsItemLayer::boundingRect() const {  
    return QRectF(0, 0, 0, 0);  
}
```

```
void QGraphicsItemLayer::paint(QPainter *, const QStyleOptionGraphicsItem *, QWidget *) {}
```

jun 26, 18 2:39

qgraphicsitemlayer.h

Page 1/1

```
#ifndef QGRAPHICSITEMLAYER_H
#define QGRAPHICSITEMLAYER_H

#include <QGraphicsItem>
#include <QObject>

class QGraphicsItemLayer : public QGraphicsItem {
public:
    QGraphicsItemLayer();

    virtual QRectF boundingRect() const;
    virtual void paint(QPainter *, const QStyleOptionGraphicsItem *, QWidget *);
};

#endif // QGRAPHICSITEMLAYER_H
```

jun 26, 18 2:39

resources.qrc

Page 1/1

```
<RCC>
  <qresource prefix="/">
    <file>assets/buttons/worm.png</file>
    <file>assets/buttons/long-girder.png</file>
    <file>assets/buttons/short-girder.png</file>
    <file>assets/stage/long_girder.png</file>
    <file>assets/stage/short_girder.png</file>
    <file>assets/stage/worm.png</file>
  </qresource>
</RCC>
```

jun 29, 18 16:28	stagedata.cpp	Page 1/3
<pre> #include "stagedata.h" #include <QDebug> #include <cassert> const qreal scale = 13.0; YAML::Emitter& operator<<(YAML::Emitter& out, const QColor& v) { out << YAML::Flow; out << YAML::BeginSeq << v.red() << v.green() << v.blue() << YAML::EndSeq; return out; } YAML::Emitter& operator<<(YAML::Emitter& out, const QPointF& v) { out << YAML::Flow; out << YAML::BeginSeq << v.x() << v.y() << YAML::EndSeq; return out; } YAML::Emitter& operator<<(YAML::Emitter& out, const WormData& v) { out << YAML::BeginMap; out << YAML::Key << "position"; out << YAML::Value << v.position; out << YAML::EndMap; return out; } YAML::Emitter& operator<<(YAML::Emitter& out, const GirderData& v) { out << YAML::BeginMap; out << YAML::Key << "position"; out << YAML::Value << v.position; out << YAML::Key << "angle"; out << YAML::Value << v.angle; out << YAML::Key << "length"; out << YAML::Value << v.length; out << YAML::EndMap; return out; } StageData::StageData(qreal width, qreal height) : width(width / scale), height(h eight / scale) {} QPointF StageData::toGameCoords(const QPointF& point) const { qreal xpos = (point.x() / scale - this->width / 2.0); qreal ypos = this->height - point.y() / scale; return QPointF(xpos, ypos); } std::size_t StageData::numWorms() const { return this->worms.size(); } void StageData::dump(std::ostream& output, std::string fileName) { YAML::Emitter emitter; emitter << YAML::BeginMap; </pre>		

jun 29, 18 16:28	stagedata.cpp	Page 2/3
<pre> emitter << YAML::Key << "name"; emitter << YAML::Value << fileName; emitter << YAML::Key << "numPlayers"; emitter << YAML::Value << this->numPlayers; emitter << YAML::Key << "weaponsAmmo"; emitter << YAML::Value << this->weapons; emitter << YAML::Key << "width"; emitter << YAML::Value << this->width; emitter << YAML::Key << "height"; emitter << YAML::Value << this->height; emitter << YAML::Key << "wormsHealth"; emitter << YAML::Value << this->wormsHealth; emitter << YAML::Key << "worms"; emitter << YAML::Value << this->worms; emitter << YAML::Key << "girders"; emitter << YAML::Value << this->girders; emitter << YAML::Key << "background"; emitter << YAML::Value; { emitter << YAML::BeginMap; emitter << YAML::Key << "closeBackgroundFile"; emitter << YAML::Value << this->closeBgFile.toStdString(); emitter << YAML::Key << "midBackgroundFile"; emitter << YAML::Value << this->medianBgFile.toStdString(); emitter << YAML::Key << "fartherBackgroundFile"; emitter << YAML::Value << this->fartherBgFile.toStdString(); emitter << YAML::Key << "color"; emitter << YAML::Value << this->bgColor; emitter << YAML::EndMap; } emitter << YAML::EndMap; assert(emitter.good()); output << emitter.c_str(); } void StageData::addWorm(QPointF position) { this->worms.push_back(WormData{this->toGameCoords(position)}); } void StageData::addShortGirder(QPointF position, qreal angle) { this->girders.push_back(GirderData{this->toGameCoords(position), -angle, 5.3 845}); } void StageData::addLongGirder(QPointF position, qreal angle) { this->girders.push_back(GirderData{this->toGameCoords(position), -angle, 10. 769}); } </pre>		

jun 29, 18 16:28

stagedata.cpp

Page 3/3

```
}  
  
void StageData::addWeaponAmmo(QString weaponName, int ammo) {  
    this->weapons[weaponName.toString()] = ammo;  
}
```

jun 29, 18 16:28

stagedata.h

Page 1/1

```

#ifndef STAGEDATA_H
#define STAGEDATA_H

#include <QColor>
#include <QDebug>
#include <QPoint>
#include <QString>
#include <Qt>
#include <iostream>
#include <vector>
#include <map>
#include "yaml-cpp/yaml.h"

struct GirderData {
    QPointF position;
    qreal angle;
    qreal length;
};

struct WormData {
    QPointF position;
};

class StageData {
public:
    QString fartherBgFile;
    QString medianBgFile;
    QString closeBgFile;
    QColor bgColor;
    int wormsHealth;
    int numPlayers;

    StageData(qreal width, qreal height);

    void dump(std::ostream &output, std::string fileName);

    std::size_t numWorms() const;

    void addWorm(QPointF position);
    void addShortGirder(QPointF position, qreal angle);
    void addLongGirder(QPointF position, qreal angle);
    void addWeaponAmmo(QString weaponName, int ammo);

private:
    QPointF toGameCoords(const QPointF &point) const;

    qreal width;
    qreal height;
    std::vector<GirderData> girders;
    std::vector<WormData> worms;
    std::map<std::string, int> weapons;
};

#endif // STAGEDATA_H

```

jun 26, 18 2:39	stageelement.cpp	Page 1/2
-----------------	-------------------------	----------

```

#include "stageelement.h"
#include <QPainter>

const double PI = 3.141592653589793;

StageElement::StageElement(const std::string &resource, ItemType type, qreal opacity)
    : QGraphicsPixmapItem(nullptr), type(type), resource(resource) {
    this->setPixmap(this->getResource(opacity));
    this->setTransformOriginPoint(this->pixmap().width() / 2, this->pixmap().height() / 2);
    this->setFlag(QGraphicsItem::ItemIsMovable);
}

ItemType StageElement::getType() {
    return this->type;
}

qreal StageElement::getAngle() const {
    return this->angle;
}

QPointF StageElement::getPosition() const {
    qreal hw = this->pixmap().width() / 2.0;
    qreal hh = this->pixmap().height() / 2.0;
    return QPointF(this->pos().x() + hw, this->pos().y() + hh);
}

void StageElement::increaseAngle() {
    this->angle += 90.0f / 10.0f;

    if (this->angle > 90.0f) {
        this->angle = 90.0f;
    }

    this->setRotation(this->angle);
}

void StageElement::decreaseAngle() {
    this->angle -= 90.0f / 10.0f;

    if (this->angle < -90.0f) {
        this->angle = -90.0f;
    }

    this->setRotation(this->angle);
}

QPixmap StageElement::getResource(qreal opacity) {
    QImage image;
    image.load(this->resource.c_str());
    image = image.convertToFormat(QImage::Format_ARGB32);

    QImage image2(image.size(), QImage::Format_ARGB32);
    image2.fill(Qt::transparent);

    QPainter painter(&image2);
    painter.setOpacity(opacity);
    painter.drawImage(image.rect(), image);

    return QPixmap::fromImage(image2);
}

```

jun 26, 18 2:39	stageelement.cpp	Page 2/2
-----------------	-------------------------	----------

```

bool StageElement::canOverlap(StageElement *) {
    return false;
}

```

jun 26, 18 2:39

stageelement.h

Page 1/1

```

#ifndef STAGEELEMENT_H
#define STAGEELEMENT_H

#include <QGraphicsItem>
#include <QGraphicsPixmapItem>
#include <QObject>
#include <QWidget>
#include <QtDebug>
#include <string>
#include "stagedata.h"

enum class ItemType {
    Worm,
    ShortGirder,
    LongGirder,
};

class StageElement : public QGraphicsPixmapItem {
public:
    StageElement(const std::string &resource, ItemType type, qreal opacity);

    ItemType getType();

    qreal getAngle() const;
    QPointF getPosition() const;

    virtual StageElement *clone() = 0;
    virtual bool canOverlap(StageElement *other);

    virtual void increaseAngle();
    virtual void decreaseAngle();

    void setRotationEnabled(bool);

    virtual void serialize(StageData &sd) = 0;

protected:
    QPixmap getResource(qreal opacity = 1.0);
    qreal angle{0.0f};
    ItemType type;

private:
    std::string resource;
};

#endif // STAGEELEMENT_H

```

jun 26, 18 2:39

stageelementworm.cpp

Page 1/1

```
#include "stageelement.h"
#include "stageelementworm.h"

StageElementWorm::StageElementWorm(qreal opacity)
    : StageElement(":/assets/stage/worm.png", ItemType::Worm, opacity) {}

void StageElementWorm::increaseAngle() {}

void StageElementWorm::decreaseAngle() {}

StageElement *StageElementWorm::clone() {
    auto *e = new StageElementWorm;
    e->angle = this->angle;
    return e;
}

void StageElementWorm::serialize(StageData &sd) {
    sd.addWorm(this->getPosition());
}
```

jun 26, 18 2:39

stageelementworm.h

Page 1/1

```
#ifndef STAGEELEMENTWORM_H
#define STAGEELEMENTWORM_H

#include "stageelement.h"

class StageElementWorm : public StageElement {
public:
    StageElementWorm(qreal opacity = 1.0);

    virtual StageElement *clone();

    virtual void increaseAngle() override;
    virtual void decreaseAngle() override;
    virtual void serialize(StageData &sd);
};

#endif // STAGEELEMENTWORM_H
```

jun 26, 18 2:39

stageelemlonggirder.cpp

Page 1/1

```
#include "stageelemlonggirder.h"

StageElemLongGirder::StageElemLongGirder(qreal opacity)
    : StageElement(":/assets/stage/long_girder.png", ItemType::LongGirder, opacity) {}

StageElement *StageElemLongGirder::clone() {
    auto *e = new StageElemLongGirder;
    e->angle = this->angle;
    e->setRotation(this->angle);
    return e;
}

void StageElemLongGirder::serialize(StageData &sd) {
    sd.addLongGirder(this->getPosition(), this->getAngle());
}

bool StageElemLongGirder::canOverlap(StageElement *other) {
    return (other->getType() != ItemType::Worm);
}
```

jun 26, 18 2:39

stageelemlonggirder.h

Page 1/1

```
#ifndef STAGEELEMlongGIRDER_H
#define STAGEELEMlongGIRDER_H

#include "stageelement.h"

class StageElemLongGirder : public StageElement {
public:
    StageElemLongGirder(qreal opacity = 1.0);

    virtual StageElement *clone();
    virtual bool canOverlap(StageElement *other);
    virtual void serialize(StageData &sd);
};

#endif // STAGEELEMlongGIRDER_H
```


jun 26, 18 2:39

stageelemshortgirder.cpp

Page 1/1

```
#include "stageelemshortgirder.h"

StageElemShortGirder::StageElemShortGirder(qreal opacity)
    : StageElement(":/assets/stage/short_girder.png", ItemType::ShortGirder, opacity) {}

StageElement *StageElemShortGirder::clone() {
    auto *e = new StageElemShortGirder;
    e->angle = this->angle;
    e->setRotation(this->angle);
    return e;
}

void StageElemShortGirder::serialize(StageData &sd) {
    sd.addShortGirder(this->getPosition(), this->getAngle());
}

bool StageElemShortGirder::canOverlap(StageElement *other) {
    return (other->getType() != ItemType::Worm);
}
```

jun 26, 18 2:39

stageelemshortgirder.h

Page 1/1

```
#ifndef STAGEELEMSHORTGIRDER_H
#define STAGEELEMSHORTGIRDER_H

#include <stageelement.h>

class StageElemShortGirder : public StageElement {
public:
    StageElemShortGirder(qreal opacity = 1.0);

    StageElement *clone();
    virtual bool canOverlap(StageElement *other);
    void serialize(StageData &sd);
};

#endif // STAGEELEMSHORTGIRDER_H
```

jun 26, 18 2:39	Animation.cpp	Page 1/3
-----------------	---------------	----------

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 17/05/18.
 */
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>

#include <algorithm>
#include <cassert>
#include <iostream>
#include <string>

#include "Animation.h"
#include "Texture.h"

GUI::Animation::Animation(const Texture &texture) : texture(&texture) {
    /* assumes the frames are squares */
    this->numFrames = this->texture->getHeight() / this->texture->getWidth();
    this->size = this->texture->getWidth();

    assert(this->numFrames > 0);
    assert(this->size > 0);
}

GUI::Animation::Animation(const Texture &texture, bool playReversed) : Animation(
texture) {
    this->playReversed = playReversed;
}

GUI::Animation::Animation(const GUI::Texture &texture, bool playReversed, int initialFrame,
                        bool autoUpdate)
    : Animation(texture, playReversed) {
    assert(this->numFrames > initialFrame);
    this->currentFrame = initialFrame;
    this->autoUpdate = autoUpdate;
}

GUI::Animation::~Animation() {}

void GUI::Animation::update(float dt) {
    if (this->autoUpdate) {
        this->elapsed += dt;
        /* checks if the frame should be updated based on the time elapsed since
the last update */
        while (this->elapsed > this->frameRate) {
            this->advanceFrame();
            this->elapsed -= this->frameRate;
        }
    }
}

/**
 * @brief Renders the animation in the given coordinates.
 *
 * @param renderer Renderer.
 * @param x X coordinate.
 * @param y Y coordinate.
 */
void GUI::Animation::render(Position &p, Camera &cam, const SDL_RendererFlip &flipType) {
    this->setFlip(flipType);

```

jun 26, 18 2:39	Animation.cpp	Page 2/3
-----------------	---------------	----------

```

    SDL_Rect clip = {0, this->size * this->currentFrame, this->size, this->size};
    cam.draw(*this->texture, p, clip, this->flipType);
}

/**
 * @brief Resets the animation.
 */
void GUI::Animation::reset() {
    this->currentFrame = 0;
}

void GUI::Animation::advanceFrame() {
    if (!this->playInverse) {
        if (this->playReversed) {
            if (this->currentFrame == 0 && this->step < 0) {
                this->step = 1;
            } else if (this->currentFrame == this->numFrames - 1 && this->step >
0) {
                this->step = -1;
            }

            if (this->playOnce &&
this->currentFrame == this->numFrames - 1) { // Por ahora ignora el
playReversed
                this->animationFinished = true;
            }

            if (!this->animationFinished) {
                this->currentFrame = (this->currentFrame + this->step) % this->numFr
ames;
            }
        } else {
            if (this->currentFrame == 0) {
                this->animationFinished = true;
            } else {
                this->currentFrame -= 1;
            }
        }
    }
}

void GUI::Animation::setFlip(SDL_RendererFlip flip_type) {
    this->flipType = flip_type;
}

SDL_RendererFlip GUI::Animation::getFlip() {
    return this->flipType;
}

void GUI::Animation::setFrame(int frame) {
    assert(frame < this->numFrames);
    assert(frame >= 0);
    this->currentFrame = frame;
}

void GUI::Animation::setAnimateOnce() {
    this->playOnce = true;
}

bool GUI::Animation::finished() {

```

jun 26, 18 2:39

Animation.cpp

Page 3/3

```
    return this->animationFinished;
}

void GUI::Animation::setAutoUpdate(bool autoUpdate) {
    this->autoUpdate = autoUpdate;
}

void GUI::Animation::setPlayInverse() {
    this->playInverse = true;
    this->currentFrame = this->numFrames - 1;
}
```

jun 26, 18 2:39	Animation.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter * Date: 17/05/18. */ #ifndef __ANIMATION_H__ #define __ANIMATION_H__ #include <SDL2/SDL.h> #include "Camera.h" #include "Color.h" #include "Exception.h" #include "Texture.h" namespace GUI { class Animation { public: Animation(const Texture &texture); Animation(const Texture &texture, bool playReversed); Animation(const Texture &texture, bool playReversed, int initialFrame, bool autoUpdate); ~Animation(); void update(float dt); void render(Position &p, Camera &cam, const SDL_RendererFlip &flipType); void reset(); void advanceFrame(); /** * Sets frame manually. Commonly used when the worm is aiming with a weapon * @param frame */ void setFrame(int frame); void setFlip(SDL_RendererFlip flipType); SDL_RendererFlip getFlip(); void setAnimateOnce(); void setAutoUpdate(bool autoUpdate); bool finished(); void setPlayInverse(); private: /** SDL texture of the raw image. */ const Texture *texture; /** * Disable Automatic update (when worm uses a bazooka, the frame changes * when user presses up or down) */ bool autoUpdate{true}; /** Current animation frame. */ int currentFrame{0}; /** Total number of frames in the sprite. */ int numFrames; /** Size of the sprite (height and width). */ int size; /** Time elapsed since last update. */ float elapsed{0.0f}; /** Frames per seconds (should this be elsewhere?). */ float frameRate{1.0f / 25.0f}; SDL_RendererFlip flipType{SDL_FLIP_NONE}; /** If true, when the animation finishes, it's played reversed instead of re starting. */ bool playReversed{false}; </pre>		

jun 26, 18 2:39	Animation.h	Page 2/2
<pre> /** If true, plays the animation once. */ bool playOnce{false}; /** Frame step. */ int step{1}; bool animationFinished{false}; bool playInverse{false}; }; } // namespace GUI #endif //__ANIMATION_H__ </pre>		

jun 29, 18 16:28	Buffer.cpp	Page 1/2
<pre> #include "Buffer.h" #include <cstdlib> #include <cstring> IO::Buffer::Buffer() { this->dataSize = 2; this->data = (uint8_t *)malloc(this->dataSize); } IO::Buffer::Buffer(const std::string &data) { this->dataSize = data.size(); this->data = (uint8_t *)malloc(this->dataSize); memcpy(this->data, data.data(), data.size()); } IO::Buffer::~~Buffer() { if (this->data) { free(this->data); } } std::string IO::Buffer::asString() { std::string output; output.reserve(this->dataSize); output.append((char *)this->data, this->dataSize); return output; } void IO::Buffer::appendFloat(float v) { static_assert(sizeof(v) == 4, "needs 32 bit float"); union { uint32_t u32; float f; } u; u.f = v; this->append(u.u32); } void IO::Buffer::appendBuffer(void *data, std::size_t dataSize) { while (this->dataSize - this->offset < dataSize) { this->grow(); } memcpy(this->data + this->offset, data, dataSize); this->offset += dataSize; } void IO::Buffer::grow() { this->dataSize *= 2; this->data = static_cast<uint8_t *>(realloc(this->data, this->dataSize)); } void IO::Buffer::getData(const uint8_t **data, std::size_t *size) { *data = this->data; *size = this->offset; } float IO::Buffer::extractFloat() { uint32_t v = this->extract<uint32_t>(); union { uint32_t u32; </pre>		

jun 29, 18 16:28	Buffer.cpp	Page 2/2
<pre> float f; } u = {v}; return u.f; } </pre>		

jun 29, 18 16:28	Buffer.h	Page 1/2
------------------	-----------------	----------

```

#ifndef BUFFER_H_
#define BUFFER_H_

#include <arpa/inet.h>
#include <stdint>
#include <cstring>
#include <string>
#include "Exception.h"

namespace IO {
class Buffer {
public:
    Buffer();
    explicit Buffer(const std::string &data);
    ~Buffer();

    template <typename NUMERIC>
    void append(NUMERIC v);
    void appendFloat(float v);
    void appendBuffer(void *data, std::size_t dataSize);

    std::string asString();

    template <typename NUMERIC>
    NUMERIC extract();
    float extractFloat();

    void getData(const uint8_t **data, std::size_t *size);

private:
    void grow();

    uint8_t *data{nullptr};
    std::size_t dataSize{0};
    std::size_t offset{0};
};
} // namespace IO

template <typename NUMERIC>
void IO::Buffer::append(NUMERIC v) {
    switch (sizeof(NUMERIC)) {
        case 1: {
            this->appendBuffer(&v, sizeof(v));
            return;
        }
        case 2: {
            uint16_t vt = htons(v);
            this->appendBuffer(&vt, sizeof(vt));
            return;
        }
        case 4: {
            uint32_t vt = htonl(v);
            this->appendBuffer(&vt, sizeof(vt));
            return;
        }
    }

    /* if it got here, the value couldn't be handled */
    throw Exception{"Invalid type"};
}

template <typename NUMERIC>

```

jun 29, 18 16:28	Buffer.h	Page 2/2
------------------	-----------------	----------

```

NUMERIC IO::Buffer::extract() {
    NUMERIC vt;
    memcpy(&vt, this->data + this->offset, sizeof(NUMERIC));

    if (this->dataSize - this->offset < sizeof(NUMERIC)) {
        throw Exception{"out of buffer"};
    }

    switch (sizeof(NUMERIC)) {
        case 1: {
            this->offset += sizeof(NUMERIC);
            return vt;
        }
        case 2: {
            this->offset += sizeof(NUMERIC);
            return ntohs(vt);
        }
        case 4: {
            this->offset += sizeof(NUMERIC);
            return ntohl(vt);
        }
    }

    /* if it got here, the value couldn't be handled */
    throw Exception{"Invalid type"};
}

#endif

```

jun 29, 18 16:28

Camera.cpp

Page 1/6

```

#include "Camera.h"
#include <cmath>
#include <iostream>

/**
 * @brief Construct a new Camera with the given initial coordinates.
 *
 * @param window The window where the camera renders.
 * @param scale The pixel/meters relation.
 * @param width The width of the area where the camera can go.
 * @param height The height of the area where the camera can go.
 */
GUI::Camera::Camera(GUI::Window &window, float scale, float width, float height)
: window(window),
  start(this->cur),
  dst(this->cur),
  scale(scale),
  renderer(window.getRenderer()),
  width(width),
  height(height) {
    this->setTo(Position{0, float(this->window.getHeight()) / this->scale / 2.0f});
}

GUI::Camera::~Camera() {}

/**
 * @brief Whether the camera is currently moving to or not.
 */
bool GUI::Camera::isMoving() const {
    return (this->dst != this->cur);
}

/**
 * @brief Returns the internal renderer.
 *
 * @return Renderer.
 */
SDL_Renderer &GUI::Camera::getRenderer() const {
    return this->renderer;
}

/**
 * @brief Returns the pixel/meters scale.
 *
 * @return Scale.
 */
float GUI::Camera::getScale() const {
    return this->scale;
}

/**
 * @brief Gets the size of the screen in game coordinates.
 *
 * @return float Screen width.
 */
float GUI::Camera::screenWidth() const {
    return float(this->window.getWidth()) / this->scale;
}

/**
 * @brief Gets the size of the screen in game coordinates.

```

jun 29, 18 16:28

Camera.cpp

Page 2/6

```

 *
 * @return float Screen height.
 */
float GUI::Camera::screenHeight() const {
    return float(this->window.getHeight()) / this->scale;
}

/**
 * @brief Returns the camera position in global coordinates.
 *
 * @return Position Camera position.
 */
GUI::Position GUI::Camera::getPosition() const {
    Position offset{(this->window.getWidth() / 2) / this->scale,
                    -(this->window.getHeight() / 2) / this->scale};
    return this->cur + offset;
}

/**
 * @brief Instantly moves the camera to the given coordinates.
 *
 * @param coord Coordinates of the new camera position.
 */
void GUI::Camera::setTo(GUI::Position coord) {
    this->elapsed = 0.0f;

    coord = this->clamp(coord);

    coord.x -= (this->window.getWidth() / 2) / this->scale;
    coord.y += (this->window.getHeight() / 2) / this->scale;
    this->dst = this->start = this->cur = coord;
}

/**
 * @brief Smoothly moves the camera to the given coordinates.
 *
 * @param coord Coordinates of the final camera position.
 */
void GUI::Camera::moveTo(GUI::Position coord) {
    coord = this->clamp(coord);

    coord.x -= (this->window.getWidth() / 2) / this->scale;
    coord.y += (this->window.getHeight() / 2) / this->scale;

    if (this->dst.distance(coord) < 7.0f) {
        this->dst = coord;
        return;
    }

    this->elapsed = 0.0f;
    this->dst = coord;
    this->start = this->cur;
}

/**
 * @brief Converts some global coordinates to screen coordinates.
 *
 * @param global Global coordinates.
 * @return Corresponding screen coordinates.
 */
GUI::ScreenPosition GUI::Camera::globalToScreen(GUI::Position global) {
    /* converts from global to local/camera coordinates */

```


jun 29, 18 16:28

Camera.cpp

Page 3/6

```

    Position local = (global - this->cur);
    local.y *= -1;

    /* calculates the screen coordinates */
    return ScreenPosition{int(local.x * this->scale), int(local.y * this->scale)};
};

/**
 * @brief Converts some screen coordinates to global coordinates.
 *
 * @param global Screen coordinates.
 * @return Corresponding global coordinates.
 */
GUI::Position GUI::Camera::screenToGlobal(GUI::ScreenPosition screen) {
    /* converts from screen to global coordinate */
    Position global = {screen.x / this->scale, screen.y / this->scale};
    global.y *= -1;
    global += this->cur;
    return global;
}

/**
 * @brief Updates the camera according to the elapsed time since the last update
 *
 * @param dt Seconds elapsed since the last update.
 */
void GUI::Camera::update(float dt) {
    if (this->dst == this->cur) {
        return;
    }

    /* total animation duration in seconds */
    const float duration = 5.0f;

    this->elapsed += dt;

    /* if the distance is less than the threshold, then it's set immediately */
    if (this->elapsed > duration || this->cur.distance(this->dst) < 0.05) {
        this->cur = this->dst;
        return;
    }

    /* calculates the new position */
    const float framerate = duration / 60.0f;
    float scale = 1.0f - 1.0f / std::pow(2, this->elapsed / framerate);
    this->cur = this->start + (this->dst - this->start) * scale;
}

void GUI::Camera::draw(const Texture &texture, Position p) {
    SDL_Rect clip = {0, 0, texture.getWidth(), texture.getHeight()};
    this->draw(texture, p, clip);
}

/**
 * @brief Draws a texture in the screen.
 *
 * @param texture The texture to render.
 * @param p Position where to draw the texture (global coordinates).
 * @param clip Portion of the texture to render.
 */

```

jun 29, 18 16:28

Camera.cpp

Page 4/6

```

void GUI::Camera::draw(const Texture &texture, Position p, const SDL_Rect &clip)
{
    this->draw(texture, p, clip, SDL_FLIP_NONE);
}

/**
 * @brief Draws a texture in the screen specifying the flip.
 *
 * @param texture The texture to render.
 * @param p Position where to draw the texture (global coordinates).
 * @param clip Portion of the texture to render.
 * @param flip Flip mode.
 * @param scale Scale factor of the final rendered texture size.
 */
void GUI::Camera::draw(const Texture &texture, Position p, const SDL_Rect &clip,
    SDL_RendererFlip flip, float scale) {
    /* converts from global to local/camera coordinates */
    Position local = (p - this->cur);
    local.y *= -1;

    /* calculates the screen coordinates */
    ScreenPosition screen_local{int(local.x * this->scale), int(local.y * this->scale)};

    /* draws in screen coordinates */
    this->drawLocal(texture, screen_local, clip, flip, scale);
}

/**
 * @brief Draws a texture in the screen in camera/screen coordinates.
 * The texture is fully rendered based in it's dimensions.
 *
 * @param texture Texture to draw.
 * @param p Position where to draw it.
 */
void GUI::Camera::drawLocal(const Texture &texture, ScreenPosition p) {
    SDL_Rect clip = {0, 0, texture.getWidth(), texture.getHeight()};
    this->drawLocal(texture, p, clip);
}

/**
 * @brief Draws a texture in the screen in camera/screen coordinates.
 *
 * @param texture The texture to render.
 * @param p Position where to draw the texture (camera coordinates).
 * @param clip Portion of the texture to render.
 */
void GUI::Camera::drawLocal(const Texture &texture, ScreenPosition p, const SDL_Rect &clip) {
    this->drawLocal(texture, p, clip, SDL_FLIP_NONE);
}

/**
 * @brief Draws a texture in the screen in camera/screen coordinates specifying the flip.
 *
 * @param texture The texture to render.
 * @param p Position where to draw the texture (camera coordinates).
 * @param clip Portion of the texture to render.
 * @param flip Flip mode.
 * @param scale Scale factor of the final texture size.
 */

```

jun 29, 18 16:28	Camera.cpp	Page 5/6
<pre> void GUI::Camera::drawLocal(const Texture &texture, ScreenPosition p, const SDL_ Rect &clip, SDL_RendererFlip flip, float scale) { /* calculates the scaled size */ int w = int(clip.w * scale); int h = int(clip.h * scale); SDL_Rect dst = {}; dst.x = p.x - w / 2; dst.y = p.y - h / 2; dst.w = w; dst.h = h; SDL_RenderCopyEx(&this->renderer, texture.get(), &clip, &dst, 0, nullptr, fl ip); } /** * @brief Clamps a given position to the camera limits. * * @param p Position to clamp. * @return Position Clamped position. */ GUI::Position GUI::Camera::clamp(Position p) const { float wWidth = this->window.getWidth() / this->scale; float wHeight = this->window.getHeight() / this->scale; /* avoids the camera from going way below the zero or above the level limits */ p.y = std::max(p.y, wHeight / 2 - 10.0f); p.y = std::min(p.y, this->height - wHeight / 2.0f); /* avoids the camera from going out of the level limits in the X axis */ p.x = std::max(p.x, (-this->width / 2.0f) + wWidth / 2.0f); p.x = std::min(p.x, (this->width / 2.0f) - wWidth / 2.0f); return p; } /** * @brief Draws a sdl rect in the screen in camera/screen coordinates specifying the flip. * * @param p Position where to draw the texture (camera coordinates). * @param clip sdl rect to render. * @param scale Scale factor of the final sdl rect size. */ void GUI::Camera::drawLocal(ScreenPosition p, const SDL_Rect &clip, SDL_Color co lor) { /* calculates the scaled size */ float scale = this->scale; int w = int(clip.w * scale); int h = int(clip.h * scale); SDL_Rect dst = {}; dst.x = p.x - w / 2; dst.y = p.y - h / 2; dst.w = w; dst.h = h; SDL_SetRenderDrawColor(&this->renderer, color.r, color.g, color.b, 0xFF); SDL_RenderFillRect(&this->renderer, &dst); </pre>		

jun 29, 18 16:28	Camera.cpp	Page 6/6
<pre> } </pre>		

jun 29, 18 16:28	Camera.h	Page 1/2
<pre> #ifndef CAMERA_H_ #define CAMERA_H_ #include <SDL2/SDL.h> #include "Point.h" #include "Texture.h" #include "Window.h" namespace GUI { using Position = Math::Point<float>; using ScreenPosition = Math::Point<int>; class Camera { public: Camera(GUI::Window &window, float scale, float width, float height); ~Camera(); bool isMoving() const; void setTo(Position coords); void moveTo(Position coords); ScreenPosition globalToScreen(Position); Position screenToGlobal(ScreenPosition); float getScale() const; Position getPosition() const; SDL_Renderer &getRenderer() const; float screenWidth() const; float screenHeight() const; void draw(const Texture &texture, Position p); void draw(const Texture &texture, Position p, const SDL_Rect &clip); void draw(const Texture &texture, Position p, const SDL_Rect &clip, SDL_RendererFlip flip, float scale = 1); void drawLocal(const Texture &texture, ScreenPosition p); void drawLocal(const Texture &texture, ScreenPosition p, const SDL_Rect &clip, SDL_RendererFlip flip, float scale = 1); void drawLocal(ScreenPosition p, const SDL_Rect &clip, SDL_Color color); Position clamp(Position p) const; void update(float dt); private: GUI::Window &window; /* current camera coordinates. */ Position cur{0, 0}; /* the position that the camera is moving towards to and the one that it started moving from. */ Position start, dst; /* Distance scale factor. */ float scale; /* elapsed time accumulator. */ float elapsed{0}; /* The SDL renderer. */ SDL_Renderer &renderer; </pre>		

jun 29, 18 16:28	Camera.h	Page 2/2
<pre> float width, height; }; } // namespace GUI #endif </pre>		

jun 26, 18 2:39

Chronometer.cpp

Page 1/1

```
#include "Chronometer.h"

Utils::Chronometer::Chronometer() {
    this->prev = std::chrono::high_resolution_clock::now();
}

Utils::Chronometer::~Chronometer() {}

/**
 * @brief Returns the number of milliseconds elapsed since the last call to this
 * function.
 *
 * @return double Milliseconds elapsed.
 */
double Utils::Chronometer::elapsed() {
    std::chrono::high_resolution_clock::time_point current =
        std::chrono::high_resolution_clock::now();

    double dt = std::chrono::duration_cast<std::chrono::duration<double>>(current
t - prev).count();
    this->prev = current;
    return dt;
}
```

jun 26, 18 2:39

Chronometer.h

Page 1/1

```
#include <chrono>

namespace Utils {
class Chronometer {
public:
    Chronometer();
    ~Chronometer();

    double elapsed();

private:
    std::chrono::high_resolution_clock::time_point prev;
};
} // namespace Utils
```

jun 26, 18 2:39

Color.h

Page 1/1

```
/*  
 * Created by Federico Manuel Gomez Peter  
 * Date: 17/05/18.  
 */
```

```
#ifndef __COLOR_H__  
#define __COLOR_H__
```

```
#include <SDL2/SDL_stdinc.h>  
namespace GUI {  
struct Color {  
    Uint8 r, g, b;  
};  
} // namespace GUI
```

```
#endif //__COLOR_H__
```

jun 26, 18 2:39

CommunicationSocket.cpp

Page 1/2

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#include <errno.h>
#include <string.h>
#include <sys/socket.h>

#include "CommunicationSocket.h"
#include "ErrorMessages.h"
#include "Exception.h"

CommunicationSocket::CommunicationSocket(int fd) : Socket(fd) {}

unsigned int CommunicationSocket::send(const char *buffer, unsigned int length)
{
    unsigned int sent = 0;
    int status = 0;

    while (sent < length) {
        status = ::send(this->fd, &buffer[sent], length - sent, MSG_NOSIGNAL);
        /*
         * Inspirado en el ejemplo del ejemplo de ayuda echoserver
         * status == 0: cerraron el socket, lanzo una excepcion de socket cerrado
         * para que no termine correctamente el servidor.
         * status < 0: hubo un error
         * status > 0: se enviaron status bytes
         */
        if (status == 0) {
            throw Exception(ERR_MSG_SOCKET_CLOSED);
        } else if (status < 0) {
            throw Exception(strerror(errno));
        } else {
            sent += status;
        }
    }

    return sent;
}

unsigned int CommunicationSocket::receive(char *buffer, unsigned int length) {
    unsigned int received = 0;
    int status = 0;

    while (received < length) {
        status = ::recv(this->fd, &buffer[received], length - received, MSG_NOSIGNAL);
        /*
         * Inspirado en el ejemplo del ejemplo de ayuda echoserver
         * status == 0: cerraron el socket, lanzo una excepcion de socket cerrado
         * para que no termine correctamente el servidor.
         * status < 0: hubo un error
         * status > 0: se enviaron status bytes
         */
        if (status == 0) {
            throw Exception(ERR_MSG_SOCKET_CLOSED);
        } else if (status < 0) {
            throw Exception(strerror(errno));
        } else {
            received += status;
        }
    }
}

```

jun 26, 18 2:39

CommunicationSocket.cpp

Page 2/2

```

    }
    return received;
}

```

jun 29, 18 16:28

CommunicationSocket.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#ifndef __COMMUNICATIONSOCKET_H__
#define __COMMUNICATIONSOCKET_H__

#include "Socket.h"
/**
 * Clase que se usa directamente en el servidor para comunicarse con el cliente.
 * Tiene la posibilidad de enviar y recibir mensajes, y es devuelta por
 * movimiento cuando se acepta una conexion.
 *
 * En el cliente, se usa indirectamente, ya que es padre de la clase
 * ClientSocket, la cual tiene la capacidad de realizar un connect al servidor.
 */
class CommunicationSocket : public Socket {
protected:
    CommunicationSocket() = default;

public:
    explicit CommunicationSocket(int fd);
    /**
     * Envia length bytes, contenidos en el buffer. Si la conexion falla, se
     * lanza una Exception. Retorna la cantidad de bytes enviados.
     * @param buffer
     * @param length
     * @return
     */
    unsigned int send(const char *buffer, unsigned int length);
    /**
     * Recibe length bytes y los guarda en el buffer. Si la conexion falla, se
     * lanza una Exception. Retorna la cantidad de bytes recibidos.
     * @param buffer
     * @param length
     * @return
     */
    unsigned int receive(char *buffer, unsigned int length);
};

#endif //__COMMUNICATIONSOCKET_H__

```


jun 26, 18 2:39

Direction.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 04/06/18
 */

#ifndef __DIRECTION_H__
#define __DIRECTION_H__

namespace Worm {
enum class Direction { right, left };
}

#endif //__DIRECTION_H__
```

jun 26, 18 2:39	DoubleBuffer.h	Page 1/2
<pre> #ifndef DOUBLEBUFFER_H_ #define DOUBLEBUFFER_H_ #include <condition_variable> #include <mutex> #include "IO.h" namespace IO { template <typename T> class DoubleBuffer { public: DoubleBuffer() {} ~DoubleBuffer() {} void swap(); void set(const T& instance); T get(bool waitNew = false); void interrupt(); private: bool hasData{false}; bool interrupted{false}; T copies[2]; int current{0}; std::mutex mutex; std::condition_variable dataSet; }; } // namespace IO /** * @brief Swaps the background copy and the current copy. */ template <typename T> void IO::DoubleBuffer<T>::swap() { std::unique_lock<std::mutex> lock(this->mutex); this->hasData = true; this->current = !this->current; this->dataSet.notify_all(); } /** * @brief Sets the background copy to the given value. * * @tparam T Instance type. * @param instance The new copy. */ template <typename T> void IO::DoubleBuffer<T>::set(const T& instance) { std::unique_lock<std::mutex> lock(this->mutex); /* sets the new value into the hidden instance */ this->copies[!this->current] = instance; } /** * @brief Gets the current copy. * * @param waitNew if true, waits until a swap was done from the last time this function was called. * if false, returns the copy eitherway. * @return The current copy. */ </pre>		

jun 26, 18 2:39	DoubleBuffer.h	Page 2/2
<pre> template <typename T> T IO::DoubleBuffer<T>::get(bool waitNew) { std::unique_lock<std::mutex> lock(this->mutex); while (!this->hasData && !this->interrupted) { this->dataSet.wait(lock); } if (this->interrupted) { throw IO::Interrupted{}; } if (waitNew) { this->hasData = false; } /* returns a copy of the visible instance */ return this->copies[this->current]; } /** * @brief Interrupts all the threads waiting in "get" launching an exception. * Useful to unblock all the observers. */ template <typename T> void IO::DoubleBuffer<T>::interrupt() { this->interrupted = true; this->dataSet.notify_all(); } #endif </pre>		

jun 26, 18 2:39

ErrorMessage.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#ifndef __ERROR_MESSAGES_H__
#define __ERROR_MESSAGES_H__

#define ERR_MSG_SOCKET_INVALID_PORT "Puerto %s inv lido: %s."
#define ERR_MSG_SOCKET_BINDING "No se pudo bindear el socket al puerto %s:"
#define ERR_MSG_SOCKET_LISTEN \
    "No se pudo establecer la cantidad de conexiones" \
    "en espera: %s."
#define ERR_MSG_SOCKET_INVALID_HOST_OR_PORT "Host %s o puerto %s inv lido: %s."
#define ERR_MSG_CONNECTION_COULD_NOT_BE_STABLISHED \
    "La conexi n a %s:%s no " \
    "pudo ser establecida."
#define ERR_MSG_SOCKET_ACCEPT "No se pudo aceptar la conexi n: %s"
#define ERR_MSG_SOCKET_CLOSED "Conexi n cerrada."
#define ERR_MSG_SOCKET_CLOSED_UNEXPECTLY \
    "Error: terminaci n inesperada al " \
    "recibir un mensaje."

#endif //__ERROR_MESSAGES_H__
```

jun 26, 18 2:39

Exception.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#include <cstdarg>
#include <sstream>
#include <string>

#include "Exception.h"
/**
 * Se adapto la clase OSError, usando funciones estandar de C++11, para hacer
 * una clase genérica Exception
 * @param fmt = Formato al cual completar
 * @param ... = argumentos para completar fmt
 */
Exception::Exception(const char *fmt, ...) noexcept {
    va_list args, args_copy;
    va_start(args, fmt);
    va_copy(args_copy, args);

    try {
        std::size_t size = std::vsnprintf(nullptr, 0, fmt, args) + 1;

        this->msg_error.reserve(size);
        std::vsnprintf(&this->msg_error.front(), size, fmt, args_copy);

        va_end(args_copy);
        va_end(args);
    } catch (...) {
        va_end(args_copy);
        va_end(args);
    }
}

const char *Exception::what() const noexcept {
    return this->msg_error.c_str();
}

```

jun 26, 18 2:39

Exception.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#ifndef __Exception_H__
#define __Exception_H__

#include <exception>
#include <string>

/**
 * Clase que se inspira de las filminas de la catedra de exception.
 * recibe en su constructor un string con el formato (al estilo del formato
 * que recibe printf), y una cantidad indefinida de parámetros adicionales,
 * para agregarselos al esqueleto de fmt.
 */
class Exception : public std::exception {
private:
    std::string msg_error;

public:
    explicit Exception(const char* fmt, ...) noexcept;
    Exception() = delete;
    virtual const char* what() const noexcept;
    virtual ~Exception() noexcept = default;
};

#endif //__Exception_H__

```

jun 26, 18 2:39

Font.cpp

Page 1/1

```
#include "Font.h"
#include "Exception.h"

GUI::Font::Font(const std::string &file, int size) : size(size) {
    this->font = TTF_OpenFont(file.c_str(), size);
    if (!this->font) {
        throw Exception{"Failed loading the font: %s", TTF_GetError()};
    }
}

GUI::Font::~Font() {
    TTF_CloseFont(this->font);
}

/**
 * @brief Gets the SDL font pointer.
 */
TTF_Font *GUI::Font::get() {
    return this->font;
}
```

jun 26, 18 2:39

Font.h

Page 1/1

```
#ifndef FONT_H_
#define FONT_H_

#include <SDL2/SDL_ttf.h>
#include <string>

namespace GUI {
class Font {
public:
    const int size;

    Font(const std::string &file, int size);
    ~Font();

    TTF_Font *get();

private:
    TTF_Font *font;
};
} // namespace GUI

#endif
```

jun 29, 18 16:28	GameStateMsg.h	Page 1/6
<pre> /* * Created by Federico Manuel Gomez Peter * Date: 17/05/18. */ #ifndef __GAME_STATE_MSG_H__ #define __GAME_STATE_MSG_H__ #define WORMS_QUANTITY 20 #define BULLETS_QUANTITY 7 #define TOTAL_TEAM_QUANTITY 5 #define WEAPONS_QUANTITY 10 #define POWER_CHARGE_TIME 5.0f #define COMMAND_GET_LEVELS 0 #define COMMAND_JOIN_GAME 1 #define COMMAND_GET_GAMES 2 #define COMMAND_CREATE_GAME 3 #define COMMAND_QUIT 5 #include "Buffer.h" #include <netinet/in.h> #include <stdint.h> #include <yaml-cpp/node/node.h> #include <yaml-cpp/yaml.h> #include <cstring> #include <vector> #include <cstdint> #include <stdint.h> #include "Direction.h" #include "Exception.h" #include "Point.h" namespace Worm { enum class StateID { Walk, Still, StartJump, Jumping, EndJump, StartBackFlip, BackFlipping, EndBackFlip, Hit, Die, Dead, Drowning, Falling, Land, Sliding, Teleporting, Teleported, Batting }; enum WeaponID { WNone, WBazooka, WGrenade, WCluster, WMortar, WBanana, </pre>		

jun 29, 18 16:28	GameStateMsg.h	Page 2/6
<pre> WHoly, WExplode, WFragment, WAerial, WDynamite, WTeleport, WBaseballBat }; } // namespace Worm namespace IO { enum class ClientGUIInput { startCreateGame, startJoinGame, quit, levelSelected, joinGame }; enum class ServerResponseAction { startGame, levelsInfo, playerConnected, gamesInfo, serverClosed }; struct ClientGUIMsg { ClientGUIMsg() = default; explicit ClientGUIMsg(ClientGUIInput input) : input(input) {}; ClientGUIInput input; }; struct ServerResponse { ServerResponseAction action; }; //TODO put this in dedicated file. enum class ServerInternalAction { lobbyFinished, quit }; struct ServerInternalMsg { ServerInternalAction action; }; struct LevelInfo { uint8_t id; std::string name; uint8_t playersQuantity; }; struct GameInfo { uint8_t gameId; uint8_t levelID; std::string levelName; uint8_t numCurrentPlayers; uint8_t numTotalPlayers; }; struct LevelData { std::string levelPath; </pre>		

jun 29, 18 16:28

GameStateMsg.h

Page 3/6

```

        std::string levelName;
        std::vector<std::string> backgroundPath;
        std::vector<std::string> backgroundName;
    };

    struct LevelsInfo : public ServerResponse {
        LevelsInfo(ServerResponseAction action, std::vector<LevelInfo> &levelsIn
fo) :
            action(action),
            levelsInfo(std::move(levelsInfo)) {}
        ServerResponseAction action;
        std::vector<LevelInfo> levelsInfo;
    };

    struct LevelSelected : public ClientGUIMsg {
        LevelSelected(ClientGUIInput input, unsigned int levelSelected) :
            ClientGUIMsg(input),
            levelSelected(levelSelected) {};
        unsigned int levelSelected;
    };

    enum class PlayerInput {
        moveNone,
        moveRight,
        moveLeft,
        startJump,
        stopMove,
        startBackFlip,
        bazooka,
        pointUp,
        pointDown,
        startShot,
        endShot,
        grenade,
        cluster,
        mortar,
        banana,
        holy,
        timeout1,
        timeout2,
        timeout3,
        timeout4,
        timeout5,
        positionSelected,
        aerialAttack,
        dynamite,
        teleport,
        baseballBat,
        disconnected
    };

    struct PlayerMsg {
        PlayerInput input;
        Math::Point<float> position{0.0f, 0.0f};

        std::string serialize() {
            IO::Buffer buffer;

            buffer.append(static_cast<uint8_t>(this->input));
            buffer.appendFloat(this->position.x);
            buffer.appendFloat(this->position.y);

            return buffer.asString();
        }
    }

```

jun 29, 18 16:28

GameStateMsg.h

Page 4/6

```

    void deserialize(const std::string &data) {
        IO::Buffer buffer{data};

        this->input = static_cast<PlayerInput>(buffer.extract<uint8_t>());
        this->position.x = buffer.extractFloat();
        this->position.y = buffer.extractFloat();
    }

    struct GameStateMsg {
        std::uint16_t elapsedTurnSeconds;
        std::int8_t windIntensity;
        std::uint8_t currentWorm;
        std::uint8_t currentWormToFollow;
        std::uint8_t currentTeam;
        std::uint8_t num_worms;
        std::uint8_t num_teams;
        std::uint8_t wormsTeam[WORMS_QUANTITY];
        Worm::Direction wormsDirection[WORMS_QUANTITY];
        std::uint16_t wormsHealth[WORMS_QUANTITY];
        std::uint32_t teamHealths[TOTAL_TEAM_QUANTITY];
        float positions[WORMS_QUANTITY * 2];
        Worm::StateID stateIDs[WORMS_QUANTITY];
        Worm::WeaponID activePlayerWeapon;
        float activePlayerAngle;
        uint8_t bulletsQuantity;
        float bullets[2 * BULLETS_QUANTITY];
        float bulletsAngle[BULLETS_QUANTITY];
        Worm::WeaponID bulletType[BULLETS_QUANTITY];
        std::int16_t weaponAmmunition[WEAPONS_QUANTITY];

        bool processingInputs;
        std::uint16_t currentPlayerTurnTime;
        bool gameEnded;
        std::uint8_t winner;
        bool playerUsedTool;
        bool waitingForNextTurn;

        std::string serialize() {
            IO::Buffer buffer;

            buffer.append(this->elapsedTurnSeconds);
            buffer.append(windIntensity);
            buffer.append(currentWorm);
            buffer.append(currentWormToFollow);
            buffer.append(currentTeam);
            buffer.append(num_worms);
            buffer.append(num_teams);

            for(std::size_t i = 0; i < this->num_worms; i++) {
                buffer.append(this->wormsTeam[i]);
                buffer.append(static_cast<uint8_t>(this->wormsDirection[i]));
                buffer.append(this->wormsHealth[i]);
                buffer.appendFloat(this->positions[i * 2]);
                buffer.appendFloat(this->positions[i * 2 + 1]);
                buffer.append(static_cast<uint8_t>(this->stateIDs[i]));
            }

            for(std::size_t i = 0; i < this->num_teams; i++) {
                buffer.append(this->teamHealths[i]);
            }
        }
    }

```

jun 29, 18 16:28

GameStateMsg.h

Page 5/6

```

    }

    buffer.append(static_cast<uint8_t>(this->activePlayerWeapon));
    buffer.appendFloat(this->activePlayerAngle);
    buffer.append(this->bulletsQuantity);

    for(std::size_t i = 0; i < this->bulletsQuantity; i++) {
        buffer.appendFloat(this->bullets[i * 2]);
        buffer.appendFloat(this->bullets[i * 2 + 1]);
        buffer.appendFloat(this->bulletsAngle[i]);
        buffer.append(static_cast<uint8_t>(this->bulletType[i]));
    }

    for(std::size_t i = 0; i < WEAPONS_QUANTITY; i++) {
        buffer.append(weaponAmmunition[i]);
    }

    buffer.append(static_cast<uint8_t>(this->processingInputs));
    buffer.append(this->currentPlayerTurnTime);
    buffer.append(static_cast<uint8_t>(this->gameEnded));
    buffer.append(this->winner);
    buffer.append(static_cast<uint8_t>(this->playerUsedTool));
    buffer.append(static_cast<uint8_t>(this->waitingForNextTurn));

    return buffer.asString();
}

void deserialize(const std::string &data) {
    IO::Buffer buffer{data};

    this->elapsedTurnSeconds = buffer.extract<uint16_t>();
    this->windIntensity = buffer.extract<uint8_t>();
    this->currentWorm = buffer.extract<uint8_t>();
    this->currentWormToFollow = buffer.extract<uint8_t>();
    this->currentTeam = buffer.extract<uint8_t>();
    this->num_worms = buffer.extract<uint8_t>();
    this->num_teams = buffer.extract<uint8_t>();

    for(std::size_t i = 0; i < this->num_worms; i++) {
        this->wormsTeam[i] = buffer.extract<uint8_t>();
        this->wormsDirection[i] = static_cast<Worm::Direction>(buffer.extrac
t<uint8_t>());
        this->wormsHealth[i] = buffer.extract<uint16_t>();
        this->positions[i * 2] = buffer.extractFloat();
        this->positions[i * 2 + 1] = buffer.extractFloat();
        this->stateIDs[i] = static_cast<Worm::StateID>(buffer.extract<uint8_
t>());
    }

    for(std::size_t i = 0; i < this->num_teams; i++) {
        this->teamHealths[i] = buffer.extract<uint32_t>();
    }

    this->activePlayerWeapon = static_cast<Worm::WeaponID>(buffer.extract<ui
nt8_t>());
    this->activePlayerAngle = buffer.extractFloat();
    this->bulletsQuantity = buffer.extract<uint8_t>();

    for(std::size_t i = 0; i < this->bulletsQuantity; i++) {
        this->bullets[i * 2] = buffer.extractFloat();
        this->bullets[i * 2 + 1] = buffer.extractFloat();
        this->bulletsAngle[i] = buffer.extractFloat();
    }

```

jun 29, 18 16:28

GameStateMsg.h

Page 6/6

```

        this->bulletType[i] = static_cast<Worm::WeaponID>(buffer.extract<uin
t8_t>());
    }

    for(std::size_t i = 0; i < WEAPONS_QUANTITY; i++) {
        this->weaponAmmunition[i] = buffer.extract<uint16_t>();
    }

    this->processingInputs = static_cast<bool>(buffer.extract<uint8_t>());
    this->currentPlayerTurnTime = buffer.extract<uint16_t>();
    this->gameEnded = static_cast<bool>(buffer.extract<uint8_t>());
    this->winner = buffer.extract<uint8_t>();
    this->playerUsedTool = static_cast<bool>(buffer.extract<uint8_t>());
    this->waitingForNextTurn = static_cast<bool>(buffer.extract<uint8_t>());
    }
};
} // namespace IO

#endif // __GAME_STATE_MSG_H__

```

jun 26, 18 2:39

IO.h

Page 1/1

```
#ifndef IO_H_
#define IO_H_

#include <exception>

namespace IO {
/**
 * @brief Interrupt exception.
 */
class Interrupted : public std::exception {
public:
    Interrupted() = default;
    ~Interrupted() = default;
};
} // namespace IO

#endif
```

jun 29, 18 16:28

Observer.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 06/06/18
 */

#ifndef __OBSERVER_H__
#define __OBSERVER_H__

enum class Event {
    Explode,
    OnExplode,
    Shot,
    Drowning,
    Drowned,
    EndTurn,
    Hit,
    EndHit,
    NewWormToFollow,
    ImpactEnd,
    TurnEnded,
    WormFalling,
    WormLanded,
    Dead,
    Dying,
    DamageOnLanding,
    NextTurn,
    Teleported,
    P2PWeaponUsed,
    StartGame,
    NewPlayer,
    EndGame,
    CreateGame,
    JoinGame,
    LevelSelected,
    ConnectionToServer,
    LobbyToJoinSelected,
    DyingDueToDisconnection,
    DeadDueToDisconnection
};

class Subject;

class Observer {
public:
    virtual ~Observer() = default;
    virtual void onNotify(Subject &subject, Event event) = 0;
};

#endif //__OBSERVER_H__

```

jun 26, 18 2:39	Point.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter * Date: 17/05/18. */ #ifndef __POINT_H__ #define __POINT_H__ #define PI 3.14159265 #include <math.h> // TODO change template for fixed class with float? namespace Math { template <typename Numeric> class Point { public: Point(Numeric x, Numeric y) : x(x), y(y) {} ~Point() {} Point<Numeric> &operator=(const Point<Numeric> &other) { this->x = other.x; this->y = other.y; return *this; } Point<Numeric> &operator+=(const Point<Numeric> &other) { this->x += other.x; this->y += other.y; return *this; } Point<Numeric> &operator-=(const Point<Numeric> &other) { this->x -= other.x; this->y -= other.y; return *this; } Point<Numeric> &operator+=(const Point<Numeric> &other) { this->x += other.x; this->y += other.y; return *this; } double distance(const Point<Numeric> &other) const { Point<Numeric> dist = *this - other; return sqrt(dist.x * dist.x + dist.y * dist.y); } friend Point<Numeric> operator/(const Point<Numeric> &p, Numeric n) { return Point<Numeric>{p.x / n, p.y / n}; } friend Point<Numeric> operator*(const Point<Numeric> &p, Numeric n) { return Point<Numeric>{p.x * n, p.y * n}; } friend bool operator==(const Point<Numeric> &a, const Point<Numeric> &b) { return ((a.x == b.x) && (a.y == b.y)); } friend bool operator!=(const Point<Numeric> &a, const Point<Numeric> &b) { return !(a == b); } friend Point<Numeric> operator+(Point<Numeric> a, const Point<Numeric> &b) { /* 'a' is passed by value! */ a += b; </pre>		

jun 26, 18 2:39	Point.h	Page 2/2
<pre> return a; } friend Point<Numeric> operator-(Point<Numeric> a, const Point<Numeric> &b) { /* 'a' is passed by value! */ a -= b; return a; } Numeric x, y; }; } // namespace Math #endif //__POINT_H__ </pre>		

jun 29, 18 16:28	Protocol.h	Page 1/4
<pre>// // Created by rodrigo on 15/06/18. // #ifndef INC_4_WORMS_PROTOCOL_H #define INC_4_WORMS_PROTOCOL_H #include <fstream> #include <netinet/in.h> #include <string> #include <vector> #include "GameStateMsg.h" #define COMMAND_LENGTH 1 #define INT_LENGTH 4 #define FILE_CHUNK_LENGTH 512 template <typename SOCKET> class Protocol { private: SOCKET socket; public: explicit Protocol(SOCKET &socket) : socket(std::move(socket)) {}; // /* Constructor por movimiento // */ // Protocol(Protocol &&protocol); // /* Termina la comunicaci3n, cerrando el socket. // */ // void stopCommunication(); /* El tipo char en este protocolo est3; asociado a los comandos. * En este caso se env3-a un comando. */ void operator<<(unsigned char command){ this->socket.send((const char *) &command, sizeof(command)); }; /* Env3-a un entero de 4 bytes sin signo en big endian. */ void operator<<(uint32_t i){ uint32_t networkInt = htonl(i); this->socket.send((char *) &networkInt, sizeof(uint32_t)); }; /* Env3-a un unsigned long trat3ndolo como un entero de 4 bytes * sin signo en big endian. */ void operator<<(unsigned long i){ *this << (unsigned int) i; }; /* Env3-a una cadena de caracteres sin el caracter de fin de cadena, * primero enviando su longitud como un entero con el m3todo ya definido. */ void operator<<(const std::string &string){ *this << (unsigned int) string.length(); this->socket.send(string.c_str(), string.length()); }; </pre>		

jun 29, 18 16:28	Protocol.h	Page 2/4
<pre>}; void operator<<(const IO::LevelInfo &levelInfo){ *this << levelInfo.id; *this << levelInfo.name; *this << levelInfo.playersQuantity; }; void operator<<(const IO::GameInfo &info) { *this << info.gameID; *this << info.levelID; *this << info.levelName; *this << info.numCurrentPlayers; *this << info.numTotalPlayers; }; template <typename T> void operator<<(const std::vector<T> &vec) { *this << vec.size(); for (const auto &elem : vec) { *this << elem; } } void operator<<(std::ifstream &file) { std::vector<char> chunk(FILE_CHUNK_LENGTH); file.seekg(0, file.end); uint32_t length = file.tellg(); file.seekg(0, file.beg); *this << length; uint32_t charactersToRead = length; while (charactersToRead > 0) { file.read(chunk.data(), FILE_CHUNK_LENGTH); uint32_t charactersRead = file.gcount(); this->socket.send(chunk.data(), charactersRead); charactersToRead -= charactersRead; } /* Recibe un comando. */ Protocol & operator>>(unsigned char &command){ char buffer; this->socket.receive(&buffer, COMMAND_LENGTH); command = buffer; return *this; }; /* Recibe un entero de 4 bytes sin signo en big endian. */ Protocol & operator>>(unsigned int &i) { std::vector<char> buffer(INT_LENGTH); this->socket.receive(buffer.data(), INT_LENGTH); i = ntohl(* (unsigned int *) buffer.data()); return *this; }; /* Recibe una cadena de caracteres recibiendo primero su longitud </pre>		

jun 29, 18 16:28

Protocol.h

Page 3/4

```

* como un entero de la forma ya especificada y luego los caracteres
* sin el caracter de fin de cadena.
*/
Protocol & operator>>(std::string &string) {
    unsigned int length;
    *this >> length;
    std::vector<char> buffer(length);
    this->socket.receive(buffer.data(), length);
    std::string localString(buffer.data(), length);
    string = std::move(localString);
    return *this;
};

Protocol & operator>>(IO::LevelInfo &levelInfo) {
    *this >> levelInfo.id;
    *this >> levelInfo.name;
    *this >> levelInfo.playersQuantity;
    return *this;
};

Protocol & operator>>(IO::GameInfo &info){
    *this >> info.gameID;
    *this >> info.levelID;
    *this >> info.levelName;
    *this >> info.numCurrentPlayers;
    *this >> info.numTotalPlayers;
    return *this;
};

template <typename T> Protocol & operator>>(std::vector<T> &vec) {
    unsigned int elements{0};
    *this >> elements;
    for (unsigned int i = 0; i < elements; i++) {
        T elem;
        *this >> elem;
        vec.emplace_back(std::move(elem));
    }

    return *this;
}

Protocol & operator>>(std::ofstream &file) {
    std::vector<char> chunk(FILE_CHUNK_LENGTH);
    uint32_t fileLength;
    *this >> fileLength;
    uint32_t fileBytesWritten = 0;

    /* La cantidad de bytes a recibir es el máximo entre los bytes
    * que quedan por recibir y el tamaño máximo de chunk.
    */
    while (fileBytesWritten < fileLength) {
        size_t bytesReceived = this->socket.receive(
            chunk.data(),
            std::min(fileLength - fileBytesWritten,
                (uint32_t) FILE_CHUNK_LENGTH));
        file.write(chunk.data(), bytesReceived);
        fileBytesWritten += bytesReceived;
    }

    return *this;
}

```

jun 29, 18 16:28

Protocol.h

Page 4/4

```

/**
 * @brief returns socket by move semantic
 * @return
 */
SOCKET getSocket() {
    this->socketRemoved = true;
    return std::move(this->socket);
};

void stopCommunication() {
    if (!this->socketRemoved) {
        this->socket.shutdown();
    }
}

bool socketRemoved{false};
};

#endif //INC_4_WORMS_PROTOCOL_H

```

jun 29, 18 16:28

Socket.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#include <cstring>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

#include "ErrorMessages.h"
#include "Exception.h"
#include "Socket.h"

Socket::Socket() : fd(-1) {}

Socket::Socket(int fd) : fd(fd) {}

Socket::Socket(Socket &&other) noexcept : fd(other.fd) {
    other.fd = -1;
}

Socket &Socket::operator=(Socket &&other) noexcept {
    if (this != &other) {
        this->fd = other.fd;
        other.fd = -1;
    }
    return *this;
}

Socket::~Socket() {
    if (this->fd != -1) {
        this->close();
    }
}

void Socket::close() {
    ::close(this->fd);
    this->fd = -1;
}

void Socket::shutdown() {
    ::shutdown(this->fd, SHUT_RDWR);
}

```


jun 26, 18 2:39

Socket.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter on 02/05/2018.
 */

#ifndef __SOCKET_H__
#define __SOCKET_H__

class Socket {
protected:
    int fd;
    /**
     * Constructor protegido, las clases hijas ClientSocket, ServerSocket y
     * DataSocket lo usaran para asignar el fd.
     */
    Socket();
    explicit Socket(int fd);
    void close();

public:
    virtual ~Socket();
    /**
     * La superclase Socket se va a poder mover, pero no copiar.
     */
    Socket(Socket &&other) noexcept;
    Socket &operator=(Socket &&other) noexcept;
    Socket(Socket &other) = delete;
    Socket &operator=(Socket &other) = delete;
    /**
     * Termina la comunicacion del socket.
     */
    void shutdown();
};

#endif //__SOCKET_H__
```

jun 26, 18 7:40

Stage.cpp

Page 1/4

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#include "Stage.h"
#include <fstream>
#include "Exception.h"
#include "GameStateMsg.h"
#include "Point.h"
#include "yaml-cpp/yaml.h"

/**
 * @brief Parses a Point from a YAML node.
 *
 * @param node Node containing the data.
 * @return Point.
 */
static Math::Point<float> _parsePoint(YAML::Node &node) {
    if (!node.IsSequence() || node.size() != 2) {
        throw Exception{"Invalid stage: Point should be a sequence of 2 floats"};
    }

    float posX = node[0].as<float>();
    float posY = node[1].as<float>();
    return Math::Point<float>{posX, posY};
}

/**
 * @brief Parses a worm's data from a YAML node.
 *
 * @param health Worm's health.
 * @param node Node containing the data.
 * @return Worm data.
 */
static Worms::WormData _parseWorm(uint16_t health, YAML::Node &node) {
    if (node.Type() != YAML::NodeType::Map) {
        throw Exception{"Invalid stage: worm data expected as Map"};
    }

    if (!node["position"]) {
        throw Exception{"Invalid stage: worm requires 'position'"};
    }

    YAML::Node positionNode = node["position"];
    return Worms::WormData{health, _parsePoint(positionNode)};
}

/**
 * @brief Parses a girder's data from a YAML node.
 *
 * @param node Node containing the data.
 * @return Girder data.
 */
static Worms::GirderData _parseGirder(YAML::Node &node) {
    if (node.Type() != YAML::NodeType::Map) {
        throw Exception{"Invalid stage: girder data expected as Map"};
    }

    if (!node["position"]) {
        throw Exception{"Invalid stage: girder requires 'position'"};
    }
}

```

jun 26, 18 7:40

Stage.cpp

Page 2/4

```

    if (!node["angle"]) {
        throw Exception{"Invalid stage: girder requires 'angle'"};
    }

    YAML::Node positionNode = node["position"];
    Math::Point<float> position = _parsePoint(positionNode);

    float angle = node["angle"].as<float>();
    float length = node["length"].as<float>();

    return Worms::GirderData{length, 1.42f, angle, position};
}

/**
 * @brief Creates a Stage object populating it from the contents of the given file.
 *
 * @param filename Name of the file with the Stage data.
 * @return Stage.
 */
Worms::Stage Worms::Stage::fromFile(const std::string &filename) {
    YAML::Node data = YAML::LoadFile(filename);

    if (!data["wormsHealth"]) {
        throw Exception{"Invalid stage: expected float 'wormsHealth'"};
    }

    uint16_t wormsHealth = data["wormsHealth"].as<unsigned short>();

    Stage stage;
    if (!data["width"] || !data["height"]) {
        throw Exception{"Invalid stage: expected stage 'width' and 'height'"};
    }
    stage.width = data["width"].as<float>();
    stage.height = data["height"].as<float>();

    if (!data["worms"] || !data["worms"].IsSequence()) {
        throw Exception{"Invalid stage: expected a sequence of 'worms'"};
    }

    if (!data["numPlayers"]) {
        throw Exception{"Invalid stage: expected integer 'numPlayers'"};
    }
    stage.numPlayers = data["numPlayers"].as<int>();

    /* loads the worms */
    for (std::size_t i = 0; i < data["worms"].size(); i++) {
        YAML::Node wormNode = data["worms"][i];
        stage.players.push_back(_parseWorm(wormsHealth, wormNode));
    }

    /* loads the girders */
    if (!data["girders"] || !data["girders"].IsSequence()) {
        throw Exception{"Invalid stage: expected a sequence of 'girders'"};
    }
    for (std::size_t i = 0; i < data["girders"].size(); i++) {
        YAML::Node girderNode = data["girders"][i];
        stage.girders.push_back(_parseGirder(girderNode));
    }

    /* loads the weapons ammo */
    YAML::Node weaponsNode;
}

```

jun 26, 18 7:40

Stage.cpp

Page 3/4

```

if (!data["weaponsAmmo"] || !data["weaponsAmmo"].IsMap()) {
    weaponsNode = YAML::LoadFile(DEFAULT_WEAPON_CONFIG_PATH);
} else {
    weaponsNode = data["weaponsAmmo"];
}

for (const auto &elem : weaponsNode) {
    auto weaponID = stage.weaponStrToID.at(elem.first.as<std::string>());
    int ammo = elem.second.as<uint16_t>();
    stage.ammunitionCounter.emplace(weaponID, ammo);
}

// -1 indicates infinite uses
stage.ammunitionCounter.emplace(Worm::WNone, -1);

/* background */
if (data["background"]) {
    YAML::Node bg = data["background"];
    if (bg["color"] && bg["color"].IsSequence()) {
        stage.backgroundColor.r = bg["color"][0].as<int>();
        stage.backgroundColor.g = bg["color"][1].as<int>();
        stage.backgroundColor.b = bg["color"][2].as<int>();
    }
    if (bg["closeBackgroundFile"]) {
        stage.closerBackgroundFile = bg["closeBackgroundFile"].as<std::string>()
;
    }
    if (bg["midBackgroundFile"]) {
        stage.midBackgroundFile = bg["midBackgroundFile"].as<std::string>();
    }
    if (bg["fartherBackgroundFile"]) {
        stage.fartherBackgroundFile = bg["fartherBackgroundFile"].as<std::string>
();
    }
}

return stage;
}

Worms::Stage::Stage() {
    this->weaponStrToID.emplace("aerialAttack", Worm::WAerial);
    this->weaponStrToID.emplace("banana", Worm::WBanana);
    this->weaponStrToID.emplace("baseballBat", Worm::WBaseballBat);
    this->weaponStrToID.emplace("bazooka", Worm::WBazooka);
    this->weaponStrToID.emplace("cluster", Worm::WCluster);
    this->weaponStrToID.emplace("dynamite", Worm::WDynamite);
    this->weaponStrToID.emplace("grenade", Worm::WGrenade);
    this->weaponStrToID.emplace("holy", Worm::WHoly);
    this->weaponStrToID.emplace("mortar", Worm::WMortar);
    this->weaponStrToID.emplace("teleport", Worm::WTeleport);
    return;
}

const std::vector<Worms::WormData> &Worms::Stage::getWorms() const {
    return this->players;
}

const std::vector<Worms::GirderData> &Worms::Stage::getGirders() const {
    return this->girders;
}

float Worms::Stage::getHeight() const {

```

jun 26, 18 7:40

Stage.cpp

Page 4/4

```

    return this->height;
}

float Worms::Stage::getWidth() const {
    return this->width;
}

const std::map<Worm::WeaponID, int16_t> &Worms::Stage::getAmmoCounter() const {
    return this->ammunitionCounter;
}

```

jun 29, 18 16:28	Stage.h	Page 1/2
<pre>/* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #ifndef __STAGE_H__ #define __STAGE_H__ #define DEFAULT_WEAPON_CONFIG_PATH "/etc/Worms/defaultWeapons.yaml" #include <stdint> #include <map> #include <string> #include <unordered_map> #include <vector> #include "GameStateMsg.h" #include "Point.h" namespace Worms { struct GirderData { float length; float height; float angle; Math::Point<float> pos; }; struct WormData { uint16_t health; Math::Point<float> position; }; struct Color { uint8_t r, g, b; }; class Stage { public: static Stage fromFile(const std::string &filename); uint8_t turnTime{10}; Color backgroundColor{255, 255, 255}; std::string fartherBackgroundFile; std::string midBackgroundFile; std::string closerBackgroundFile; int numPlayers; Stage(); ~Stage() = default; const std::vector<WormData> &getWorms() const; const std::vector<GirderData> &getGirders() const; float getHeight() const; float getWidth() const; const std::map<Worm::WeaponID, std::int16_t> &getAmmoCounter() const; private: std::vector<WormData> players; std::vector<GirderData> girders; std::unordered_map<std::string, Worm::WeaponID> weaponStrToID; std::map<Worm::WeaponID, std::int16_t> ammunitionCounter;</pre>		

jun 29, 18 16:28	Stage.h	Page 2/2
<pre> float width{100.0f}; float height{30.0f}; }; } // namespace Worms #endif //__STAGE_H__</pre>		

jun 26, 18 2:39

Stream.h

Page 1/2

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 17/05/18.
 */

#ifndef __STREAM_H__
#define __STREAM_H__

#include <atomic>
#include <condition_variable>
#include <queue>

#include "Exception.h"

namespace IO {
template <typename Msg>
class Stream {
public:
    Stream() {}
    Stream(Stream &&other) = default;
    Stream &operator=(Stream &&other) = default;
    Stream &operator=(Stream &other) = delete;

    ~Stream() {
        this->close();
    }

    void push(const Msg &m) {
        std::unique_lock<std::mutex> lock(this->mutex);
        this->q.push(m);
        this->notEmpty.notify_all();
    }

    bool pop(Msg &m, bool block = true) {
        std::unique_lock<std::mutex> lock(this->mutex);
        while (this->q.empty() && !this->closed) {
            if (!block) {
                return false;
            }
            this->notEmpty.wait(lock);
        }

        if (this->closed) {
            throw Exception{"closed"};
        }

        m = this->q.front();
        this->q.pop();
        return true;
    }

    Stream &operator<<(const Msg &m) {
        this->push(m);
        return *this;
    }

    Stream &operator>>(Msg &m) {
        this->pop(m);
        return *this;
    }

    void close() {

```

jun 26, 18 2:39

Stream.h

Page 2/2

```

        if (this->closed) {
            return;
        }
        this->closed = true;
        this->notEmpty.notify_all();
    }

private:
    std::queue<Msg> q;
    std::mutex mutex;
    std::condition_variable notEmpty;
    std::atomic<bool> closed{false};
};
} // namespace IO
#endif //__STREAM_H__

```

jun 29, 18 16:28

Subject.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 06/06/18
 */

#include "Subject.h"

void Subject::addObserver(Observer *obs) {
    this->observers.emplace(obs);
}
// TODO check possibly raceCondition
void Subject::removeObserver(Observer *obs) {
    this->observers.erase(this->observers.find(obs));
}

void Subject::notify(Subject &subject, Event event) {
    for (auto &observer : this->observers) {
        observer->onNotify(subject, event);
    }
}
```

jun 29, 18 16:28

Subject.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 06/06/18
 */

#ifndef __Subject_H__
#define __Subject_H__

#include <set>

#include "Observer.h"

class Subject {
public:
    Subject() = default;
    Subject(Subject &copy) = delete;
    virtual ~Subject() = default;
    void addObserver(Observer *obs);
    void removeObserver(Observer *obs);
    /**
     * Notify all observers with the Event id, so the Observer can do
     * what is necessary
     * @param subject
     * @param event
     */
    void notify(Subject &subject, Event event);

protected:
    std::set<Observer *> observers;
    int numObservers;
};

#endif //__Subject_H__
```

jun 29, 18 16:28	Text.cpp	Page 1/2
------------------	----------	----------

```

#include "Text.h"
#include <random>
#include "Exception.h"

std::unordered_map<GUI::TextCacheKey, GUI::TexturePtr, GUI::TextKeyHash> GUI::Text::cache;

GUI::Text::Text(GUI::Font& font) : font(font) {}

GUI::Text::~Text() {}

void GUI::Text::setBackground(SDL_Color color) {
    this->hasBackground = true;
    this->background = color;
    this->needs_render = true;
}

void GUI::Text::set(const std::string& text, SDL_Color color) {
    this->set(text, color, this->font.size);
}

void GUI::Text::set(const std::string& text, SDL_Color color, int size) {
    this->size = size;
    this->text = text;
    this->color = color;
    this->needs_render = true;
}

void GUI::Text::render(GUI::Position p, GUI::Camera& camera) {
    if (this->needs_render) {
        this->createTexture(&camera.getRenderer());
        this->needs_render = false;
    }

    float scale = float(this->size) / float(this->font.size);

    if (this->hasBackground) {
        GUI::ScreenPosition sp = camera.globalToScreen(p);

        SDL_Renderer& renderer = camera.getRenderer();
        SDL_SetRenderDrawColor(&renderer, this->background.r, this->background.g,
            this->background.b, 255);

        SDL_Rect r = {};
        r.x = sp.x - this->texture->getWidth() * scale / 2;
        r.y = sp.y - (this->texture->getHeight() * scale - 10) / 2;
        r.w = this->texture->getWidth() * scale;
        r.h = this->texture->getHeight() * scale - 10;

        SDL_RenderFillRect(&renderer, &r);
    }

    SDL_Rect clip = {0, 0, this->texture->getWidth(), this->texture->getHeight()};
    camera.draw(*this->texture, p, clip, SDL_FLIP_NONE, scale);
}

/**
 * @brief Renders the text in a screen coordintates.
 *
 * @param p Position where the text will be rendered in screen coordintaes.

```

jun 29, 18 16:28	Text.cpp	Page 2/2
------------------	----------	----------

```

 * @param camera Camera.
 */
void GUI::Text::renderFixed(GUI::ScreenPosition p, GUI::Camera& camera) {
    this->render(camera.screenToGlobal(p), camera);
}

/**
 * @brief Creates a texture for the current text.
 *
 * @param renderer Renderer.
 */
void GUI::Text::createTexture(SDL_Renderer* renderer) {
    /* checks if the texture is already in the cache */
    uint32_t color = 0 | this->color.r << 16 | this->color.g << 8 | this->color.b;

    TextCacheKey key(this->text, color);
    if (GUI::Text::cache.find(key) != GUI::Text::cache.end()) {
        this->texture = GUI::Text::cache.at(key);
        return;
    }

    /* if the cache is full, removes a random entry */
    if (Text::cache.size() >= 150) {
        std::mt19937 rng;
        rng.seed(std::random_device()());
        std::uniform_int_distribution<std::mt19937::result_type> r(0, Text::cache.size() - 1);
        auto random_it = std::next(std::begin(Text::cache), r(rng));
        Text::cache.erase(random_it);
    }

    /* render text surface */
    SDL_Surface* surface = TTF_RenderText_Solid(this->font.get(), text.c_str(), this->color);
    if (surface == NULL) {
        throw Exception{"Failed rendering text surface: %s", TTF_GetError()};
    }

    /* convert to a texture */
    SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, surface);
    if (texture == NULL) {
        SDL_FreeSurface(surface);
        throw Exception{"Failed create texture from rendered text: %s", SDL_GetError()};
    }

    GUI::Text::cache[key] = TexturePtr(new Texture{texture, surface->w, surface->h});
    this->texture = GUI::Text::cache[key];

    SDL_FreeSurface(surface);
}

```


jun 26, 18 7:40

Text.h

Page 1/1

```

#ifndef TEXT_H_
#define TEXT_H_

#include <SDL2/SDL.h>
#include <cstdint>
#include <functional>
#include <list>
#include <memory>
#include <string>
#include <unordered_map>
#include "Camera.h"
#include "Font.h"

namespace GUI {
using TextCacheKey = std::pair<std::string, uint32_t>;

struct TextKeyHash {
    size_t operator()(const TextCacheKey &p) const {
        return std::hash<std::string>{}(p.first) ^ p.second;
    }
};

using TexturePtr = std::shared_ptr<Texture>;

class Text {
public:
    Text(Font &font);
    ~Text();

    void set(const std::string &text, SDL_Color color);
    void set(const std::string &text, SDL_Color color, int size);
    void setBackground(SDL_Color color);

    void render(Position p, Camera &camera);
    void renderFixed(ScreenPosition p, Camera &camera);

protected:
    static std::unordered_map<TextCacheKey, TexturePtr, TextKeyHash> cache;

private:
    void createTexture(SDL_Renderer *renderer);

    bool hasBackground{false};
    SDL_Color background;
    /* internal texture to represent the text in screen. */
    TexturePtr texture;
    /* text content to render. */
    std::string text;
    /* whether the texture should be rendered. */
    bool needs_render{true};
    /* the font to use. */
    Font &font;
    SDL_Color color;
    int size{-1};
};
} // namespace GUI

#endif

```

jun 26, 18 2:39	Texture.cpp	Page 1/2
-----------------	--------------------	----------

```

#include "Texture.h"
#include <SDL2/SDL_image.h>
#include <cassert>
#include "Exception.h"

GUI::Texture::Texture(const std::string &filename, SDL_Renderer &renderer, GUI::
Color key) {
    /* loads the image into a temporary surface */
    SDL_Surface *tmp = IMG_Load(filename.c_str());
    if (!tmp) {
        throw Exception{"Error loading %s: %s", filename.c_str(), IMG_GetError()};
    }

    SDL_SetColorKey(tmp, SDL_TRUE, SDL_MapRGB(tmp->format, key.r, key.g, key.b))
;

    /* creates a texture from the surface */
    this->texture = SDL_CreateTextureFromSurface(&renderer, tmp);
    if (!this->texture) {
        SDL_FreeSurface(tmp);
        throw Exception{"Error creating texture for %s: %s", filename.c_str(), SDL_GetErro
r()};
    }

    this->height = tmp->h;
    this->width = tmp->w;

    /* releases the temporary surface */
    SDL_FreeSurface(tmp);
}

GUI::Texture::Texture(SDL_Texture *texture, int width, int height)
: height(height), width(width), texture(texture) {
    if (!texture) {
        throw Exception{"Texture cannot be NULL"};
    }
}

/**
 * @brief Move constructor.
 *
 * @param other Instance to move.
 */
GUI::Texture::Texture(Texture &&other) {
    this->height = other.height;
    this->width = other.width;
    std::swap(this->texture, other.texture);
}

GUI::Texture::~Texture() {
    if (this->texture) {
        SDL_DestroyTexture(this->texture);
    }
}

/**
 * @brief Gets the texture's width.
 *
 * @return int width.
 */
int GUI::Texture::getWidth() const {
    return this->width;

```

jun 26, 18 2:39	Texture.cpp	Page 2/2
-----------------	--------------------	----------

```

}

/**
 * @brief Gets the texture's height.
 *
 * @return int Height.
 */
int GUI::Texture::getHeight() const {
    return this->height;
}

/**
 * @brief Returns the internal SDL texture.
 *
 * @return SDL texture.
 */
SDL_Texture *GUI::Texture::get() const {
    return this->texture;
}

```

jun 26, 18 2:39

Texture.h

Page 1/1

```
#ifndef TEXTURE_H_
#define TEXTURE_H_

#include <SDL2/SDL.h>
#include <string>
#include "Color.h"

namespace GUI {
class Texture {
public:
    Texture(const std::string &filename, SDL_Renderer &renderer, Color key);
    Texture(SDL_Texture *texture, int width, int height);
    Texture(Texture &&other);
    ~Texture();

    int getWidth() const;
    int getHeight() const;
    SDL_Texture *get() const;

private:
    int height{0}, width{0};
    SDL_Texture *texture{nullptr};
};
} // namespace GUI

#endif
```

jun 26, 18 2:39

TextureManager.h

Page 1/1

```

#ifndef TEXTURE_MANAGER_H_
#define TEXTURE_MANAGER_H_

#include <SDL2/SDL.h>
#include <functional>
#include <string>
#include <unordered_map>
#include "Texture.h"

namespace GUI {
template <typename ID, typename HASH = std::hash<ID>>
class TextureManager {
public:
    TextureManager(SDL_Renderer& renderer);
    ~TextureManager();
    TextureManager& operator=(TextureManager& other) = delete;

    void load(ID id, const std::string& file_name, Color key);
    const Texture& get(ID id) const;

private:
    std::unordered_map<ID, Texture, HASH> cache;
    SDL_Renderer& renderer;
};
} // namespace GUI

template <typename ID, typename HASH>
GUI::TextureManager<ID, HASH>::TextureManager(SDL_Renderer& renderer) : renderer
(renderer) {}

template <typename ID, typename HASH>
GUI::TextureManager<ID, HASH>::~~TextureManager() {}

/**
 * @brief Loads a texture.
 *
 * @param file_name The image file name.
 * @param renderer Renderer.
 */
template <typename ID, typename HASH>
void GUI::TextureManager<ID, HASH>::load(ID id, const std::string& file_name, GU
I::Color key) {
    GUI::Texture texture(file_name, this->renderer, key);
    this->cache.insert(std::make_pair(id, std::move(texture)));
}

/**
 * @brief Gets a texture.
 *
 * @param file_name Name of the texture.
 */
template <typename ID, typename HASH>
const GUI::Texture& GUI::TextureManager<ID, HASH>::get(ID id) const {
    return this->cache.at(id);
}

#endif

```

jun 29, 18 16:28

Thread.cpp

Page 1/1

```
//  
// Created by rodrigo on 15/06/18.  
//  
  
#include <thread>  
#include "Thread.h"  
  
Thread::Thread(Thread &&thread) noexcept {  
    this->thread = std::move(thread.thread);  
}  
  
Thread & Thread::operator=(Thread &&thread) noexcept {  
    this->thread = std::move(thread.thread);  
    return *this;  
}  
  
void Thread::start() {  
    this->thread = std::thread(&Thread::run, this);  
}  
  
void Thread::join() {  
    this->thread.join();  
}
```

jun 29, 18 16:28

Thread.h

Page 1/1

```

//
// Created by rodrigo on 15/06/18.
//

#ifndef INC_4_WORMS_THREAD_H
#define INC_4_WORMS_THREAD_H

#include <thread>

class Thread {
private:
    std::thread thread;
public:
    Thread() = default;

    /* El thread del objeto recibido de asigna al propio
     * mediante move semantics.
     */
    Thread(Thread &&thread) noexcept;

    /* El thread del objeto recibido de asigna al propio
     * mediante move semantics.
     */
    Thread & operator=(Thread &&thread) noexcept;

    /* Se elimina el constructor copia ya que no debe usarse para un thread.
     */
    Thread(const Thread&) = delete;

    /* Se elimina el operador asignaciÃ³n ya que no debe usarse para un thread.
     */
    Thread & operator=(const Thread&) = delete;

    /* Comienza la ejecuciÃ³n del hilo.
     */
    void start();

    /* Hace el join del hilo.
     */
    void join();

    /* Este mÃ©todo deben implementarlo todas las clases que hereden de esta,
     * ya que es el mÃ©todo que el thread ejecutarÃ¡.
     */
    virtual void run() = 0;

    /* Este mÃ©todo deben implementarlo todas las clases que hereden de esta,
     * ya que debe terminar la ejecuciÃ³n del hilo de una forma ordenada.
     */
    virtual void stop() = 0;

    /* El destructor es el default.
     */
    virtual ~Thread() = default;
};

#endif //INC_4_WORMS_THREAD_H

```

jun 26, 18 2:39

utils.h

Page 1/1

```
#ifndef UTILS_H_
#define UTILS_H_

namespace Utils {

/**
 * @brief This class allows using enums classes as hash keys.
 *
 */
struct EnumClassHash {
    template <typename T>
    std::size_t operator()(T t) const {
        return static_cast<std::size_t>(t);
    }
};
} // namespace Utils

#endif
```

jun 26, 18 2:39	Window.cpp	Page 1/3
<pre> /* * Created by Federico Manuel Gomez Peter * Date: 17/05/18. */ #include <SDL2/SDL.h> #include <SDL2/SDL_image.h> #include <SDL2/SDL_mixer.h> #include <SDL2/SDL_ttf.h> #include <iostream> #include "Exception.h" #include "Window.h" GUI::Window::Window() : Window(WINDOW_WIDTH, WINDOW_HEIGHT) {} GUI::Window::Window(int width, int height) { if (SDL_Init(SDL_INIT EVERYTHING) < 0) { throw Exception{"SDL could not initialize: %s", SDL_GetError()}; } if (!SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "1")) { std::cerr << "Warning: Linear texture filtering not enabled!" << std::endl; } this->window = SDL_CreateWindow("Worms", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, width, height, SDL_WINDOW_SHOWN SDL_WINDOW_RESIZABLE); if (!this->window) { this->close(); throw Exception{"Window could not be created: %s", SDL_GetError()}; } this->renderer = SDL_CreateRenderer(this->window, -1, SDL_RENDERER_ACCELERATED SDL_RENDERER_PRESENTVSYNC); if (!this->renderer) { this->close(); throw Exception{"Failed creating the render: %s", SDL_GetError()}; } if ((!IMG_Init(IMG_INIT_PNG)) & IMG_INIT_PNG) { this->close(); throw Exception{"Failed initialiing IMG: %s", IMG_GetError()}; } if (TTF_Init() == -1) { this->close(); throw Exception{"SDL_ttf could not initialize! SDL_ttf Error: %s", TTF_GetError()}; } SDL_SetWindowSize(this->window, width, height); if (Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048) < 0) { throw Exception{"SDL_mixer could not initialize! SDL_mixer Error: %s\n", Mix_GetError()}; } } GUI::Window::~Window() { this->close(); </pre>		

jun 26, 18 2:39	Window.cpp	Page 2/3
<pre> } /** * @brief Maximizes the window. */ void GUI::Window::maximize() { SDL_MaximizeWindow(this->window); } void GUI::Window::close() { if (this->renderer != nullptr) { SDL_DestroyRenderer(this->renderer); this->renderer = nullptr; } if (this->window != nullptr) { SDL_DestroyWindow(this->window); this->window = nullptr; } Mix_Quit(); TTF_Quit(); IMG_Quit(); SDL_Quit(); } void GUI::Window::clear() { this->clear({0xFF, 0xFF, 0xFF, 0xFF}); } void GUI::Window::clear(SDL_Color color) { SDL_SetRenderDrawColor(this->renderer, color.r, color.g, color.b, color.a); SDL_RenderClear(this->renderer); } void GUI::Window::render() { SDL_RenderPresent(this->renderer); } SDL_Renderer& GUI::Window::getRenderer() { return *this->renderer; } /** * @brief Gets the window's height. */ int GUI::Window::getHeight() const { int h, w; SDL_GL_GetDrawableSize(this->window, &w, &h); return h; } /** * @brief Gets the window's width. */ int GUI::Window::getWidth() const { int h, w; </pre>		

jun 26, 18 2:39

Window.cpp

Page 3/3

```
    SDL_GL_GetDrawableSize(this->window, &w, &h);  
    return w;  
}  
  
/**  
 * @brief Checks if the mouse cursor is contained in this window.  
 *  
 * @return true is the mouse is in the current window.  
 */  
bool GUI::Window::containsMouse() const {  
    return (SDL_GetMouseFocus() == this->window);  
}
```

jun 26, 18 2:39

Window.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 17/05/18.
 */

#ifndef __Window_H__
#define __Window_H__

#include <SDL2/SDL.h>

#define WINDOW_WIDTH 1200
#define WINDOW_HEIGHT 800

namespace GUI {
class Window {
public:
    Window();
    Window(int width, int height);
    ~Window();

    void maximize();
    void clear();
    void clear(SDL_Color color);
    void render();
    SDL_Renderer &getRenderer();

    int getWidth() const;
    int getHeight() const;
    bool containsMouse() const;

private:
    SDL_Window *window{nullptr};
    SDL_Renderer *renderer{nullptr};
    void close();
};
} // namespace GUI

#endif //__Window_H__

```

jun 26, 18 2:39

WrapTexture.cpp

Page 1/2

```
#include "WrapTexture.h"

GUI::WrapTexture::WrapTexture(const GUI::Texture& texture, float width, float height)
    : texture(texture), width(width), height(height) {}

GUI::WrapTexture::~WrapTexture() {}

/**
 * @brief Renders the texture wrapped.
 *
 * @param p Position where to render.
 * @param camera Camera.
 */
void GUI::WrapTexture::render(GUI::Position p, Camera& camera) {
    this->render(p, 0.0f, camera);
}

/**
 * @brief Renders the texture wrapped with the given angle.
 *
 * @param p Position where to render.
 * @param angle Render angle.
 * @param camera Camera.
 */
void GUI::WrapTexture::render(GUI::Position p, float angle, Camera& camera) {
    int width = this->width * camera.getScale();
    int height = this->height * camera.getScale();

    int cols = width / this->texture.getWidth();
    int rows = height / this->texture.getHeight();

    int x_remainder = width % this->texture.getWidth();
    int y_remainder = height % this->texture.getHeight();

    ScreenPosition sp = camera.globalToScreen(p);
    sp.x -= width / 2;
    sp.y -= height / 2;

    SDL_Renderer* renderer = &camera.getRenderer();
    int tw = this->texture.getWidth();
    int th = this->texture.getHeight();

    for (int i = 0; i < cols; i++) {
        for (int j = 0; j < rows; j++) {
            ScreenPosition pos{sp.x + tw * i, sp.y - th * j};
            ScreenPosition rcenter = sp - pos;
            SDL_Point center = {rcenter.x + width / 2, rcenter.y + height / 2};

            SDL_Rect dst = {pos.x, pos.y, tw, th};
            SDL_RenderCopyEx(renderer, this->texture.get(), NULL, &dst, -angle,
&center,
                                SDL_FLIP_NONE);
        }

        if (y_remainder) {
            ScreenPosition pos{sp.x + tw * i, sp.y - th * rows};
            ScreenPosition rcenter = sp - pos;
            SDL_Point center = {rcenter.x + width / 2, rcenter.y + height / 2};

            SDL_Rect clip = {0, 0, tw, y_remainder};
            SDL_Rect dst = {pos.x, pos.y, tw, y_remainder};

```

jun 26, 18 2:39

WrapTexture.cpp

Page 2/2

```
SDL_RenderCopyEx(renderer, this->texture.get(), &clip, &dst, -angle,
&center,
                                SDL_FLIP_NONE);
        }
    }

    if (x_remainder > 0) {
        for (int i = 0; i < rows; i++) {
            ScreenPosition pos{sp.x + tw * cols, sp.y - th * i};
            ScreenPosition rcenter = sp - pos;
            SDL_Point center = {rcenter.x + width / 2, rcenter.y + height / 2};

            SDL_Rect clip = {0, 0, x_remainder, th};
            SDL_Rect dst = {pos.x, pos.y, x_remainder, th};
            SDL_RenderCopyEx(renderer, this->texture.get(), &clip, &dst, -angle,
&center,
                                SDL_FLIP_NONE);
        }

        if (y_remainder) {
            ScreenPosition pos{sp.x + tw * cols, sp.y - th * rows};
            ScreenPosition rcenter = sp - pos;
            SDL_Point center = {rcenter.x + width / 2, rcenter.y + height / 2};

            SDL_Rect clip = {0, 0, x_remainder, y_remainder};
            SDL_Rect dst = {pos.x, pos.y, x_remainder, y_remainder};
            SDL_RenderCopyEx(renderer, this->texture.get(), &clip, &dst, -angle,
&center,
                                SDL_FLIP_NONE);
        }
    }
}

void GUI::WrapTexture::renderFixed(GUI::ScreenPosition sp, Camera& camera) {
    sp.x -= this->width * camera.getScale() / 2;
    sp.y -= this->height * camera.getScale() / 2;
    this->render(camera.screenToGlobal(sp), camera);
}
```

jun 26, 18 2:39

WrapTexture.h

Page 1/1

```
#ifndef WRAP_TEXTURE_H_
#define WRAP_TEXTURE_H_

#include "Camera.h"
#include "Texture.h"

namespace GUI {
class WrapTexture {
public:
    WrapTexture(const Texture &texture, float width, float height);
    ~WrapTexture();

    void render(Position p, Camera &camera);
    void render(Position p, float angle, Camera &camera);
    void renderFixed(ScreenPosition p, Camera &camera);

private:
    const Texture &texture;
    float width, height;
};
} // namespace GUI

#endif
```

jun 26, 18 2:39

AerialAttack.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#include "AerialAttack.h"
#define CONFIG Game::Config::getInstance()

Weapon::AerialAttack::AerialAttack()
: Weapon::Weapon(CONFIG.getAerialAttackConfig(), Worm::WeaponID::WAerial, 0.
0),
    bulletsQuantity(CONFIG.getAerialAttackMissileQuantity()),
    missileSeparation(CONFIG.getAerialAttackMissileSeparation()) {}

void Weapon::AerialAttack::update(float dt) {}

void Weapon::AerialAttack::startShot(Worms::Player *player) {}

void Weapon::AerialAttack::endShot() {}

void Weapon::AerialAttack::setTimeout(uint8_t time) {}

std::list<Worms::Bullet> Weapon::AerialAttack::onExplode(const Worms::Bullet &mainBullet,
                                                         Worms::Physics &physics
) {
    return std::move(std::list<Worms::Bullet>());
}

void Weapon::AerialAttack::positionSelected(Worms::Player &p, Math::Point<float>
point) {
    point.y += CONFIG.getAerialAttackLaunchHeight();
    point.x -= this->missileSeparation * (this->bulletsQuantity + 1) / 2;

    Worms::BulletInfo bulletInfo = {this->config.dmgInfo,
                                     point,
                                     this->config.minAngle,
                                     (float)this->config.maxShotPower,
                                     0,
                                     this->config.restitution,
                                     this->config.friction,
                                     this->config.explosionInitialTimeout,
                                     Event::Explode,
                                     this->config.bulletRadius,
                                     this->config.bulletDampingRatio,
                                     this->config.windAffected};

    std::list<Worms::Bullet> ret;
    for (int i = 0; i < this->bulletsQuantity; i++) {
        point.x += this->missileSeparation;
        bulletInfo.point = point;
        ret.emplace_back(bulletInfo, p.getPhysics(), Worm::WeaponID::WAerial);
    }

    p.endShot(ret);
}

```

jun 26, 18 2:39

AerialAttack.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#ifndef __AERIAL_ATTACK_H__
#define __AERIAL_ATTACK_H__

#include "../Player.h"
#include "Weapon.h"

namespace Weapon {
class AerialAttack : public Worms::Weapon {
public:
    AerialAttack();
    ~AerialAttack() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                      Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    const uint8_t bulletsQuantity{0};
    const float missileSeparation{0};
};
} // namespace Weapon

#endif //__AERIAL_ATTACK_H__

```

jun 26, 18 2:39	BackFlipping.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 21/05/18 */ #include "BackFlipping.h" #include "../Player.h" Worms::BackFlipping::BackFlipping(GUI::Position p) : State(Worm::StateID::BackFlipping), startPosition(p) {} void Worms::BackFlipping::update(Worms::Player &p, float dt, b2Body *body) { /* * when the worm lands (there was a collision between the worm and the * girder) it has to changes its state to endJump, and take an impulse * of equal absolute value and different sign of the impulse taken in * startJump stage (remember, the worm has a friction coefficient 0). * * In the y-axis there will be no impulse because its velocity was * cancelled because of the collision with the girder. */ this->timeElapsed += dt; if (p.isOnGround()) { float32 mass = body->GetMass(); b2Vec2 previousVel = body->GetLinearVelocity(); b2Vec2 impulses = {mass * (0.0f - previousVel.x), 0.0f}; body->ApplyLinearImpulseToCenter(impulses, true); p.landDamage(this->startPosition.y - p.getPosition().y); p.setState(Worm::StateID::Land); // p.setState(Worm::StateID::EndBackFlip); } } void Worms::BackFlipping::moveRight(Worms::Player &p) {} void Worms::BackFlipping::moveLeft(Worms::Player &p) {} void Worms::BackFlipping::jump(Worms::Player &p) {} void Worms::BackFlipping::stopMove(Worms::Player &p) {} void Worms::BackFlipping::backFlip(Worms::Player &p) {} void Worms::BackFlipping::bazooka(Worms::Player &p) {} void Worms::BackFlipping::pointUp(Worms::Player &p) {} void Worms::BackFlipping::pointDown(Worms::Player &p) {} void Worms::BackFlipping::startShot(Worms::Player &p) {} void Worms::BackFlipping::endShot(Worms::Player &p) {} void Worms::BackFlipping::grenade(Worms::Player &p) {} void Worms::BackFlipping::cluster(Worms::Player &p) {} void Worms::BackFlipping::mortar(Worms::Player &p) {} void Worms::BackFlipping::banana(Worms::Player &p) {} </pre>		

jun 26, 18 2:39	BackFlipping.cpp	Page 2/2
<pre> void Worms::BackFlipping::holy(Worms::Player &p) {} void Worms::BackFlipping::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::BackFlipping::aerialAttack(Worms::Player &p) {} void Worms::BackFlipping::dynamite(Worms::Player &p) {} void Worms::BackFlipping::teleport(Worms::Player &p) {} void Worms::BackFlipping::baseballBat(Worms::Player &p) {} </pre>		

jun 26, 18 2:39

BackFlipping.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#ifndef __PLAYER_BACK_FLIPPING_H__
#define __PLAYER_BACK_FLIPPING_H__

#include <Camera.h>
#include <stdint>
#include "PlayerState.h"

namespace Worms {
class BackFlipping : public State {
public:
    BackFlipping(GUI::Position p);
    ~BackFlipping() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    GUI::Position startPosition;
};
} // namespace Worms

#endif //__PLAYER_BACK_FLIPPING_H__

```


jun 26, 18 2:39

Banana.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#include "Banana.h"
#include "../Player.h"

Weapon::Banana::Banana(float angle)
    : Worms::Weapon(Game::Config::getInstance().getBananaConfig(), Worm::WeaponI
D::WBanana, angle) {
    this->powerChargeTime = Game::Config::getInstance().getPowerChargeTime();
}

void Weapon::Banana::update(float dt) {
    if (this->increaseShotPower) {
        if (this->shotPower < this->config.maxShotPower) {
            this->shotPower += dt / this->powerChargeTime * this->config.maxShot
Power;
        }
    }
}

void Weapon::Banana::startShot(Worms::Player *player) {
    this->increaseShotPower = true;
}

void Weapon::Banana::endShot() {
    this->increaseShotPower = false;
    this->shotPower = 0;
}

void Weapon::Banana::setTimeout(uint8_t time) {
    this->timeLimit = time;
}

std::list<Worms::Bullet> Weapon::Banana::onExplode(const Worms::Bullet &mainBull
et,
    Worms::Physics &physics) {
    return std::move(std::list<Worms::Bullet>());
}

void Weapon::Banana::positionSelected(Worms::Player &p, Math::Point<float> point
) {}

```

jun 26, 18 2:39

Banana.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#ifndef __Banana_H__
#define __Banana_H__

#include "Weapon.h"

namespace Weapon {
class Banana : public Worms::Weapon {
public:
    Banana(float angle);
    ~Banana() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                     Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    float powerChargeTime{0.0f};
};
} // namespace Weapon

#endif // __Banana_H__

```

jun 26, 18 2:39

BaseballBat.cpp

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#include "BaseballBat.h"
#include "../Player.h"
#include "Direction.h"

Weapon::BaseballBat::BaseballBat(float angle)
: Worms::Weapon(Game::Config::getInstance().getBaseballBatConfig(),
  Worm::WeaponID::WBaseballBat, angle),
  weaponInfo{this->config.dmgInfo, Worm::Direction::left, {0, 0}} {
  this->isP2P = true;
}

void Weapon::BaseballBat::update(float dt) {}

void Weapon::BaseballBat::startShot(Worms::Player *player) {
  this->weaponInfo.position = player->getPosition();
  this->weaponInfo.direction = player->direction;
  this->weaponInfo.angle = this->angle;
}

void Weapon::BaseballBat::endShot() {}

void Weapon::BaseballBat::setTimeout(uint8_t time) {}

std::list<Worms::Bullet> Weapon::BaseballBat::onExplode(const Worms::Bullet &mainBullet,
  Worms::Physics &physics)
{
  return std::move(std::list<Worms::Bullet>());
}

void Weapon::BaseballBat::positionSelected(Worms::Player &p, Math::Point<float> point) {}

Config::P2PWeapon &Weapon::BaseballBat::getWeaponInfo() {
  return this->weaponInfo;
}

```

jun 26, 18 2:39

BaseballBat.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_BASEBALLBAT_H
#define INC_4_WORMS_BASEBALLBAT_H

#include "../Config/P2PWeapon.h"
#include "../Physics.h"
#include "Weapon.h"

namespace Weapon {
class BaseballBat : public Worms::Weapon {
public:
    BaseballBat(float angle);
    ~BaseballBat() = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                      Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;
    Config::P2PWeapon &getWeaponInfo();

private:
    Config::P2PWeapon weaponInfo;
};
} // namespace Weapon

#endif // INC_4_WORMS_BASEBALLBAT_H

```

jun 26, 18 7:40

Batting.cpp

Page 1/1

```

//
// Created by rodrigo on 23/06/18.
//

#include "Batting.h"
#include "../Config/Config.h"
#include "../Player.h"

Worms::Batting::Batting()
    : State(Worm::StateID::Batting), battingTime(Game::Config::getInstance().getBattingTime()) {}

void Worms::Batting::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed >= this->battingTime) {
        p.setState(Worm::StateID::Still);
    }
}

void Worms::Batting::moveRight(Worms::Player &p) {}
void Worms::Batting::moveLeft(Worms::Player &p) {}
void Worms::Batting::jump(Worms::Player &p) {}
void Worms::Batting::stopMove(Worms::Player &p) {}
void Worms::Batting::backFlip(Worms::Player &p) {}
void Worms::Batting::bazooka(Worms::Player &p) {}
void Worms::Batting::pointUp(Worms::Player &p) {}
void Worms::Batting::pointDown(Worms::Player &p) {}
void Worms::Batting::startShot(Worms::Player &p) {}
void Worms::Batting::endShot(Worms::Player &p) {}
void Worms::Batting::grenade(Worms::Player &p) {}
void Worms::Batting::cluster(Worms::Player &p) {}
void Worms::Batting::mortar(Worms::Player &p) {}
void Worms::Batting::banana(Worms::Player &p) {}
void Worms::Batting::holy(Worms::Player &p) {}
void Worms::Batting::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::Batting::aerialAttack(Worms::Player &p) {}
void Worms::Batting::dynamite(Worms::Player &p) {}
void Worms::Batting::teleport(Worms::Player &p) {}
void Worms::Batting::baseballBat(Worms::Player &p) {}

```

jun 26, 18 7:40

Batting.h

Page 1/1

```

//
// Created by rodrigo on 23/06/18.
//

#ifndef INC_4_WORMS_BATTING_H
#define INC_4_WORMS_BATTING_H

#include "PlayerState.h"

namespace Worms {
class Batting : public State {
public:
    Batting();
    ~Batting() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    float battingTime;
};
}

#endif // INC_4_WORMS_BATTING_H

```

jun 26, 18 2:39

Bazooka.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#include "Bazooka.h"
#include "../Player.h"

Weapon::Bazooka::Bazooka(float angle)
    : Worms::Weapon(Game::Config::getInstance().getBazookaConfig(), Worm::Weapon
ID::WBazooka,
                    angle) {
    this->powerChargeTime = Game::Config::getInstance().getPowerChargeTime();
}

void Weapon::Bazooka::update(float dt) {
    if (this->increaseShotPower) {
        if (this->shotPower < this->config.maxShotPower) {
            this->shotPower += dt / this->powerChargeTime * this->config.maxShot
Power;
        } else {
            this->player->endShot();
        }
    }
}

void Weapon::Bazooka::startShot(Worms::Player *player) {
    this->increaseShotPower = true;
    this->player = player;
}

void Weapon::Bazooka::endShot() {
    this->increaseShotPower = false;
    this->shotPower = 0;
}

void Weapon::Bazooka::setTimeout(uint8_t time) {}

std::list<Worms::Bullet> Weapon::Bazooka::onExplode(const Worms::Bullet &mainBul
let,
                                                    Worms::Physics &physics) {
    return std::move(std::list<Worms::Bullet>());
}

void Weapon::Bazooka::positionSelected(Worms::Player &p, Math::Point<float> poin
t) {}

```

jun 26, 18 2:39

Bazooka.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#ifndef __BAZOOKA_H__
#define __BAZOOKA_H__

#include "Weapon.h"

namespace Weapon {
class Bazooka : public Worms::Weapon {
public:
    Bazooka(float angle);
    ~Bazooka() = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                     Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    float powerChargeTime{0.0f};
    Worms::Player *player;
};
} // namespace Weapon

#endif // __BAZOOKA_H__

```


jun 26, 18 2:39

BulletConfig.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 22/06/18
 */

#include "BulletConfig.h"
#include "ConfigDefines.h"

Config::Bullet::DamageInfo::DamageInfo(const YAML::Node &config)
: damage((std::uint16_t)config[DAMAGE].as<unsigned int>()),
  radius(config[RADIUS].as<float>()),
  impulseDampingRatio(config[IMPULSE_DAMPING_RATIO].as<float>()) {}
```

jun 26, 18 7:40

BulletConfig.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 22/06/18
 */

#ifndef __BULLET_CONFIG_H__
#define __BULLET_CONFIG_H__

#include <cstdint>
#include "yaml-cpp/yaml.h"

namespace Config {
namespace Bullet {
struct DamageInfo {
    std::uint16_t damage;
    float radius;
    float impulseDampingRatio;

    explicit DamageInfo(const YAML::Node &config);
};
} // namespace Bullet
} // namespace Bullet

#endif //__BULLET_CONFIG_H__
```

jun 29, 18 16:28	Bullet.cpp	Page 1/3
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 26/05/18 */ #include <cmath> #include <iostream> #include "../Config/Config.h" #include "Bullet.h" #include "Weapon.h" #include "../Physics.h" #include "../PhysicsEntity.h" Worms::Bullet::Bullet(BulletInfo &info, Worms::Physics &physics, Worm::WeaponID weapon) : PhysicsEntity(Worms::EntityID::EtBullet), physics(physics), weaponID(weapo n), info(info) { float distance = info.safeNonContactDistance + info.radius; this->bodyDef.type = b2_dynamicBody; this->bodyDef.position.Set(info.point.x + distance * cos(info.angle * PI / 1 80.0f), info.point.y + distance * sin(info.angle * PI / 1 80.0f)); this->bodyDef.fixedRotation = true; this->body = this->physics.createBody(this->bodyDef); this->shape.m_p.Set(0.0f, 0.0f); this->shape.m_radius = info.radius; this->fixture.shape = &this->shape; this->fixture.density = 1.0f; this->fixture.restitution = info.restitution; this->fixture.friction = info.friction; this->body->CreateFixture(&this->fixture); this->body->SetUserData(this); // this->body->SetTransform(this->body->GetPosition(), info.angle); } void Worms::Bullet::update(float dt, Config::Wind wind) { if (this->keepUpdating) { this->timeElapsed += dt; if (!this->impulseApplied) { float32 mass = this->body->GetMass(); b2Vec2 impulses = {mass * float32(this->info.power * this->info.damp ingRatio * cos(this->info.angle * PI / 180.0f)), mass * float32(this->info.power * this->info.damp ingRatio * sin(this->info.angle * PI / 180.0f))}; b2Vec2 position = this->body->GetWorldCenter(); this->body->ApplyLinearImpulse(impulses, position, true); this->impulseApplied = true; } else { b2Vec2 velocity = this->body->GetLinearVelocity(); this->info.angle = atan2(velocity.y, velocity.x) * 180.0f / PI; if (this->info.angle < 0) { this->info.angle += 360.0f; } } } } </pre>		

jun 29, 18 16:28	Bullet.cpp	Page 2/3
<pre> } if (this->info.windAffected) { this->body->ApplyForceToCenter(b2Vec2{wind.instensity * wind.xDirect ion, 0.0f}, true); } if (this->hasExploded()) { this->notify(*this, this->info.explodeEvent); this->weaponID = Worm::WeaponID::WExplode; this->keepUpdating = false; b2Vec2 lastP = this->body->GetPosition(); this->lastPosition = {lastP.x, lastP.y}; this->destroyBody(); } } Math::Point<float> Worms::Bullet::getPosition() const { if (this->keepUpdating) { b2Vec2 p = this->body->GetPosition(); return Math::Point<float>(p.x, p.y); } else { return this->lastPosition; } } float Worms::Bullet::getAngle() const { return (this->info.angle >= 0 && this->info.angle < 90) ? this->info.angle + 360.0f : this->info.angle; } void Worms::Bullet::startContact(Worms::PhysicsEntity *physicsEntity) { this->madeImpact = true; } void Worms::Bullet::endContact(Worms::PhysicsEntity *physicsEntity) {} Worms::Bullet::~Bullet() { this->destroyBody(); } bool Worms::Bullet::hasExploded() const { if (this->getPosition().y < Game::Config::getInstance().getWaterLevel()) { return true; } if (this->info.explotionTimeout > 0) { return this->timeElapsed >= this->info.explotionTimeout; } else { return this->madeImpact; } } Config::Bullet::DamageInfo Worms::Bullet::getDamageInfo() const { return this->info.dmgInfo; } bool Worms::Bullet::operator<(Worms::Bullet &other) { return this->timeElapsed > other.timeElapsed; } </pre>		

jun 29, 18 16:28

Bullet.cpp

Page 3/3

```
Worm::WeaponID Worms::Bullet::getWeaponID() const {  
    return this->weaponID;  
}  
  
void Worms::Bullet::destroyBody() {  
    if (this->body != nullptr) {  
        this->body->GetWorld()->DestroyBody(this->body);  
        this->body = nullptr;  
    }  
}
```

jun 29, 18 16:28	Bullet.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 26/05/18 */ #ifndef __BULLET_H__ #define __BULLET_H__ #include <GameStateMsg.h> #include "../Config/Config.h" #include "../Config/WindConfig.h" #include "../libs/Observer.h" #include "../Physics.h" #include "../PhysicsEntity.h" #include "Point.h" namespace Worms { struct BulletInfo { Config::Bullet::DamageInfo dmgInfo; Math::Point<float> point; float angle; float power; float safeNonContactDistance; float restitution; float friction; uint8_t explotionTimeout; Event explodeEvent; float radius; float dampingRatio; bool windAffected; }; /** * forward declaration of weapon. */ class Weapon; class Bullet : public PhysicsEntity { public: Bullet(BulletInfo &i, Worms::Physics &physics, Worm::WeaponID weaponID); ~Bullet(); /** * Apply initial impulse in the first iteration, or estimate the * bullet's tangential velocity to guide the animation. Finally, checks if * an Explode event occurred, and notify his observer if so. * @param dt * @param w */ void update(float dt, Config::Wind wind); Math::Point<float> getPosition() const; float getAngle() const; /** * Sets its impact boolean to true. Usefull for detecting explosion in * bullets that explode on first impact. * @param physicsEntity */ virtual void startContact(Worms::PhysicsEntity *physicsEntity) override; virtual void endContact(Worms::PhysicsEntity *physicsEntity) override; /** * return true if the bullet is under the water, if its timeout (in the * case that it have it) has been reached, or if it has collided with * something * @return </pre>		

jun 29, 18 16:28	Bullet.h	Page 2/2
<pre> */ bool hasExploded() const; Config::Bullet::DamageInfo getDamageInfo() const; bool operator<(Worms::Bullet &other); Worm::WeaponID getWeaponID() const; private: b2Body *body{nullptr}; b2BodyDef bodyDef; b2CircleShape shape; b2FixtureDef fixture; Worms::Physics &physics; bool impulseApplied{false}; float timeElapsed{0.0f}; bool madeImpact{false}; Worm::WeaponID weaponID; BulletInfo info; bool keepUpdating{true}; Math::Point<float> lastPosition{0, 0}; void destroyBody(); }; // namespace Worms #endif //__BULLET_H__ </pre>		

jun 26, 18 2:39	Cluster.cpp	Page 1/2
-----------------	--------------------	----------

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#include "Cluster.h"

#define CONFIG Game::Config::getInstance()

Weapon::Cluster::Cluster(float angle)
: Worms::Weapon(CONFIG.getClusterConfig(), Worm::WeaponID::WCluster, angle),
  fragmentConfig(CONFIG.getClusterFragmentConfig()) {
    this->powerChargeTime = CONFIG.getPowerChargeTime();
}

void Weapon::Cluster::update(float dt) {
    if (this->increaseShotPower) {
        if (this->shotPower < this->config.maxShotPower) {
            this->shotPower += dt / this->powerChargeTime * this->config.maxShot
Power;
        }
    }
}

void Weapon::Cluster::startShot(Worms::Player *player) {
    this->increaseShotPower = true;
}

void Weapon::Cluster::endShot() {
    this->increaseShotPower = false;
    this->shotPower = 0;
}

void Weapon::Cluster::setTimeout(uint8_t time) {
    this->timeLimit = time;
}

std::list<Worms::Bullet> Weapon::Cluster::onExplode(const Worms::Bullet &mainBul
let,
                                                    Worms::Physics &physics) {
    uint8_t fragmentQuantity = CONFIG.getClusterFragmentQuantity();
    Math::Point<float> p = mainBullet.getPosition();
    Worms::BulletInfo bulletInfo = {this->fragmentConfig.dmgInfo,
                                    p,
                                    this->fragmentConfig.minAngle,
                                    (float)this->fragmentConfig.maxShotPower,
                                    this->fragmentConfig.bulletRadius * 6,
                                    this->fragmentConfig.restitution,
                                    this->fragmentConfig.friction,
                                    this->fragmentConfig.explotionInitialTimeout
    },
    Event::Explode,
    this->fragmentConfig.bulletRadius,
    this->fragmentConfig.bulletDampingRatio,
    this->config.windAffected};

    std::list<Worms::Bullet> ret;
    for (int i = 0; i < fragmentQuantity; i++) {
        bulletInfo.angle = i * this->fragmentConfig.angleStep + this->fragmentCo
nfig.minAngle;
        ret.emplace_back(bulletInfo, physics, Worm::WeaponID::WFragment);
    }

```

jun 26, 18 2:39	Cluster.cpp	Page 2/2
-----------------	--------------------	----------

```

        return std::move(ret);
    }

void Weapon::Cluster::positionSelected(Worms::Player &p, Math::Point<float> poin
t) {}

```

jun 26, 18 2:39

Cluster.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#ifndef __CLUSTER_H__
#define __CLUSTER_H__

#include "../Physics.h"
#include "../Player.h"
#include "Weapon.h"

namespace Weapon {
class Cluster : public Worms::Weapon {
public:
    Cluster(float angle);
    ~Cluster() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                     Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    const Config::Weapon &fragmentConfig;
    float powerChargeTime{0.0f};
};
} // namespace Weapon

#endif //__CLUSTER_H__

```

jun 29, 18 16:28	Config.cpp	Page 1/4
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 01/06/18 */ #include "Config.h" #include <iostream> #include "ConfigDefines.h" #include "yaml-cpp/yaml.h" /** * Meyer's singleton implementation. * @return */ Game::Config &Game::Config::getInstance() { static Config instance(YAML::LoadFile(CONFIG_PATH)); return instance; } Game::Config::Config(const YAML::Node &node) : jumpVelocity(node[JUMP][VELOCITY][X].as<float>(), node[JUMP][VELOCITY][Y].as<float>()), backflipVelocity(node[BACKFLIP][VELOCITY][X].as<float>(), node[BACKFLIP][VELOCITY][Y].as<float>()), startJumpTime(node[JUMP][START_TIME].as<float>()), landTime(node[JUMP][LAND_TIME].as<float>()), walkVelocity(node[WALK][VELOCITY].as<float>()), safeFallDistance(node[GAME][SAFE_FALL_DISTANCE].as<float>()), maxFallDamage(node[GAME][MAX_FALL_DAMAGE].as<float>()), turnTime((std::uint8_t)node[GAME][TURN_TIME].as<unsigned int>()), extraTurnTime(node[GAME][EXTRA_TURN_TIME].as<float>()), waitForNextTurnTime(node[GAME][WAIT_FOR_NEXT_TURN_TIME].as<float>()), powerChargeTime(node[GAME][POWER_CHARGE_MAX_TIME].as<float>()), dyingTime(node[GAME][DYING_TIME].as<float>()), drowningTime(node[GAME][DROWNING_TIME].as<float>()), battingTime(node[GAME][BATTING_TIME].as<float>()), teleportTime(node[GAME][TELEPORT_TIME].as<float>()), waterLevel(node[GAME][WATER_LEVEL].as<int>()), minWindIntensity(node[WIND_INTENSITY][MIN].as<float>()), maxWindIntensity(node[WIND_INTENSITY][MAX].as<float>()), bazooka(node[BAZOOKA]), greenGrenade(node[GRENADE]), cluster(node[CLUSTER]), clusterFragments(node[CLUSTER][FRAGMENT]), clusterFragmentQuantity((std::uint8_t)node[CLUSTER][FRAGMENT][QUANTITY].as<unsigned int>()), mortar(node[MORTAR]), mortarFragments(node[MORTAR][FRAGMENT]), mortarFragmentQuantity((std::uint8_t)node[MORTAR][FRAGMENT][QUANTITY].as<unsigned int>()), banana(node[BANANA]), holy(node[HOLY]), aerialAttackMissileQuantity((std::uint8_t)node[AERIAL_ATTACK][BULLET][QUANTITY].as<unsigned int>()), aerialAttackMissileSeparation(node[AERIAL_ATTACK][BULLET][SEPARATION].as<float>()), aerialAttack(node[AERIAL_ATTACK]), aerialAttackLaunchHeight(node[AERIAL_ATTACK][LAUNCH_HEIGHT].as<float>()), dynamite(node[DYNAMITE]), teleport(node[TELEPORT]), </pre>		

jun 29, 18 16:28	Config.cpp	Page 2/4
<pre> baseballBat(node[BASEBALL_BAT]) {} float Game::Config::getSafeFallDistance() const { return this->safeFallDistance; } float Game::Config::getMaxFallDamage() const { return this->maxFallDamage; } const Math::Vector Game::Config::getJumpVelocity() const { return this->jumpVelocity; } const float Game::Config::getStartJumpTime() const { return this->startJumpTime; } const float Game::Config::getLandTime() const { return this->landTime; } const Math::Vector Game::Config::getBackflipVelocity() const { return this->backflipVelocity; } const uint8_t Game::Config::getTurnTime() const { return this->turnTime; } const float Game::Config::getGameWidth() const { return this->gameWidth; } const float Game::Config::getGameHeight() const { return this->gameHeight; } const uint16_t Game::Config::getWormHealth() const { return this->wormHealth; } const Config::Weapon &Game::Config::getBazookaConfig() const { return this->bazooka; } const float Game::Config::getDyingTime() const { return this->dyingTime; } const float Game::Config::getDrowningTime() const { return this->drowningTime; } const float Game::Config::getExtraTurnTime() const { return this->extraTurnTime; } const int Game::Config::getWaterLevel() const { return this->waterLevel; } </pre>		

jun 29, 18 16:28	Config.cpp	Page 3/4
<pre> const float Game::Config::getWalkVelocity() const { return this->walkVelocity; } const Config::Weapon &Game::Config::getGreenGrenadeConfig() const { return this->greenGrenade; } const Config::Weapon &Game::Config::getClusterConfig() const { return this->cluster; } const Config::Weapon &Game::Config::getMortarConfig() const { return this->mortar; } const Config::Weapon &Game::Config::getBananaConfig() const { return this->banana; } const Config::Weapon &Game::Config::getHolyConfig() const { return this->holy; } const float Game::Config::getPowerChargeTime() const { return this->powerChargeTime; } const Config::Weapon &Game::Config::getClusterFragmentConfig() const { return this->clusterFragments; } const uint8_t Game::Config::getClusterFragmentQuantity() const { return this->clusterFragmentQuantity; } const Config::Weapon &Game::Config::getMortarFragmentConfig() const { return this->mortarFragments; } const uint8_t Game::Config::getMortarFragmentQuantity() const { return this->mortarFragmentQuantity; } const float Game::Config::getWaitForNextTurnTime() const { return this->waitForNextTurnTime; } const Config::Weapon &Game::Config::getAerialAttackConfig() const { return this->aerialAttack; } const uint8_t Game::Config::getAerialAttackMissileQuantity() const { return this->aerialAttackMissileQuantity; } const float Game::Config::getAerialAttackMissileSeparation() const { return this->aerialAttackMissileSeparation; } const float Game::Config::getAerialAttackLaunchHeight() const { return this->aerialAttackLaunchHeight; </pre>		

jun 29, 18 16:28	Config.cpp	Page 4/4
<pre> } const float Game::Config::getBattingTime() const { return this->battingTime; } const Config::Weapon &Game::Config::getTeleportConfig() const { return this->teleport; } const float Game::Config::getTeleportTime() const { return this->teleportTime; } const Config::Weapon &Game::Config::getDynamiteConfig() const { return this->dynamite; } const Config::Weapon &Game::Config::getBaseballBatConfig() const { return this->baseballBat; } float Game::Config::getMinWindIntensity() const { return this->minWindIntensity; } float Game::Config::getMaxWindIntensity() const { return this->maxWindIntensity; } </pre>		

jun 29, 18 16:28	ConfigDefines.h	Page 1/2
<pre>/* * Created by Federico Manuel Gomez Peter. * date: 22/06/18 */ #ifndef __CONFIG_DEFINES_H__ #define __CONFIG_DEFINES_H__ #define CONFIG_PATH "/etc/Worms/serverConfig.yaml" #define JUMP "jump" #define VELOCITY "velocity" #define X "x" #define Y "y" #define BACKFLIP "backflip" #define START_TIME "startTime" #define LAND_TIME "landTime" #define WALK "walk" #define GAME "game" #define SAFE_FALL_DISTANCE "safeFallDistance" #define MAX_FALL_DAMAGE "maxFallDamage" #define TURN_TIME "turnTime" #define EXTRA_TURN_TIME "extraTurnTime" #define WAIT_FOR_NEXT_TURN_TIME "waitForNextTurnTime" #define POWER_CHARGE_MAX_TIME "powerChargeMaxTime" #define DYING_TIME "dyingTime" #define DROWNING_TIME "drowningTime" #define BATTING_TIME "battingTime" #define TELEPORT_TIME "teleportTime" #define WATER_LEVEL "waterLevel" #define WIND_INTENSITY "windIntensity" #define MIN "min" #define MAX "max" #define BAZOOKA "bazooka" #define GRENADE "grenade" #define CLUSTER "cluster" #define FRAGMENT "fragment" #define QUANTITY "quantity" #define MORTAR "mortar" #define BANANA "banana" #define HOLY "holy" #define AERIAL_ATTACK "aerialAttack" #define BULLET "bullet" #define SEPARATION "separation" #define LAUNCH_HEIGHT "launchHeight" #define DYNAMITE "dynamite" #define TELEPORT "teleport" #define BASEBALL_BAT "baseballBat" #define DAMAGE "damage" #define RADIUS "radius" #define IMPULSE_DAMPING_RATIO "impulseDampingRatio" #define ANGLE "angle" #define STEP "step" #define MAX_SHOT_POWER "maxShotPower" #define RESTITUTION "restitution" #define FRICTION "friction" #define EXPLOSION_INITIAL_TIMEOUT "explosionInitialTimeout" #define HAS_AFTER_EXPLODE "hasAfterExplode" #define DAMPING_RATIO "dampingRatio" #define WIND_AFFECTED "windAffected"</pre>		

jun 29, 18 16:28	ConfigDefines.h	Page 2/2
<pre>#endif //__CONFIG_DEFINES_H__</pre>		

jun 26, 18 7:45	Config.h	Page 1/3
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 01/06/18 */ #ifndef __GAMECONFIG_H__ #define __GAMECONFIG_H__ #include <stdint.h> #include <mutex> #include "Direction.h" #include "Point.h" #include "WeaponConfig.h" #include "WindConfig.h" #define NUM_TEAMS 2 #define GAME_HEIGHT 30.0f #define GAME_WIDTH 30.0f #define WORM_HEALTH 100 namespace Math { using Vector = Math::Point<float>; } namespace Game { /** * Singleton class with all the game configuration (Velocity constants, * Weapons attributes, etc) */ class Config { public: static Config &getInstance(); ~Config() = default; const Math::Vector getJumpVelocity() const; const Math::Vector getBackflipVelocity() const; const float getStartJumpTime() const; const float getLandTime() const; const float getWalkVelocity() const; float getSafeFallDistance() const; float getMaxFallDamage() const; float getMinWindIntensity() const; float getMaxWindIntensity() const; const uint8_t getTurnTime() const; const float getExtraTurnTime() const; const float getWaitForNextTurnTime() const; const float getPowerChargeTime() const; const float getGameWidth() const; const float getGameHeight() const; const float getDyingTime() const; const float getDrowningTime() const; const float getBattingTime() const; const float getTeleportTime() const; const int getWaterLevel() const; const uint16_t getWormHealth() const; const ::Config::Weapon &getBazookaConfig() const; const ::Config::Weapon &getGreenGrenadeConfig() const; </pre>		

jun 26, 18 7:45	Config.h	Page 2/3
<pre> const uint8_t getClusterFragmentQuantity() const; const ::Config::Weapon &getClusterConfig() const; const ::Config::Weapon &getMortarConfig() const; const ::Config::Weapon &getBananaConfig() const; const ::Config::Weapon &getHolyConfig() const; const ::Config::Weapon &getClusterFragmentConfig() const; const ::Config::Weapon &getMortarFragmentConfig() const; const uint8_t getMortarFragmentQuantity() const; const ::Config::Weapon &getAerialAttackConfig() const; const ::Config::Weapon &getDynamiteConfig() const; const uint8_t getAerialAttackMissileQuantity() const; const float getAerialAttackMissileSeparation() const; const float getAerialAttackLaunchHeight() const; const ::Config::Weapon &getTeleportConfig() const; const ::Config::Weapon &getBaseballBatConfig() const; private: /** * Constructor hidden because is a singleton. * TODO change constructor so it loads information from yaml file */ // Config(); explicit Config(const YAML::Node &node); Config(Config &copy) = delete; Config(Config &&other) = delete; Config &operator=(Config &copy) = delete; Config &operator=(Config &&other) = delete; // jump const Math::Vector jumpVelocity; const Math::Vector backflipVelocity; const float startJumpTime; const float landTime; // moving const float walkVelocity; // game const float safeFallDistance; const float maxFallDamage; const std::uint8_t turnTime; const float extraTurnTime; const float waitForNextTurnTime; const float powerChargeTime; uint8_t numTeams(NUM_TEAMS); float gameWidth(GAME_WIDTH); float gameHeight(GAME_HEIGHT); uint16_t wormHealth(WORM_HEALTH); const float dyingTime; const float drowningTime; const float battingTime; const float teleportTime; const int waterLevel; const float minWindIntensity; const float maxWindIntensity; // weapons const ::Config::Weapon bazooka; const ::Config::Weapon greenGrenade; const ::Config::Weapon cluster; const ::Config::Weapon clusterFragments; const uint8_t clusterFragmentQuantity; const ::Config::Weapon mortar; const ::Config::Weapon mortarFragments; </pre>		

jun 26, 18 7:45

Config.h

Page 3/3

```
const uint8_t mortarFragmentQuantity;
const ::Config::Weapon banana;
const ::Config::Weapon holy;
const uint8_t aerialAttackMissileQuantity;
const float aerialAttackMissileSeparation;
const ::Config::Weapon aerialAttack;
const float aerialAttackLaunchHeight;
const ::Config::Weapon dynamite;
const ::Config::Weapon teleport;
const ::Config::Weapon baseballBat;
};

void endTurn();
} // namespace Game

#endif // __GAMECONFIG_H__
```

jun 29, 18 16:28	ContactEventListener.cpp	Page 1/2
------------------	---------------------------------	----------

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/05/18
 */

#include <iostream>

#include "ContactEventListener.h"
#include "Player.h"

/**
 * @brief Pre collision solver handler for Box2D. Notifies colliding objects so
the can act
 * appropriately.
 * @param contact Collision contact.
 * @param oldManifold Manifold.
 */
void ContactEventListener::PreSolve(b2Contact *contact, const b2Manifold *oldManifold) {
    Worms::PhysicsEntity *e1 =
        static_cast<Worms::PhysicsEntity *>(contact->GetFixtureA()->GetBody()->GetUserData());
    Worms::PhysicsEntity *e2 =
        static_cast<Worms::PhysicsEntity *>(contact->GetFixtureB()->GetBody()->GetUserData());

    if (!e1 || !e2) {
        return;
    }

    e1->contactWith(*e2, *contact);
    e2->contactWith(*e1, *contact);
}

void ContactEventListener::BeginContact(b2Contact *contact) {
    Worms::PhysicsEntity *playerA =
        static_cast<Worms::PhysicsEntity *>(contact->GetFixtureA()->GetBody()->GetUserData());
    Worms::PhysicsEntity *playerB =
        static_cast<Worms::PhysicsEntity *>(contact->GetFixtureB()->GetBody()->GetUserData());
    /**
     * If fixture A is a Worm, then call startContact. This will delegate
     * the action to the internal state. For example, when a worm jump,
     * it run with a state startJump, after a few seconds (so the clients
     * could animate the impulse the worm takes to jump), it changes its
     * state to Jumping. The moment the state changes to endJump will be
     * when box2d detects a collision between the worm and the girder.
     */
    if (playerA) {
        playerA->startContact(playerB);
    }
    if (playerB) {
        playerB->startContact(playerA);
    }

    void *fixtureData = contact->GetFixtureA()->GetUserData();
    if (fixtureData) {
        Worms::PhysicsEntity *sensor = static_cast<Worms::TouchSensor *>(fixtureData);
        sensor->startContact(playerB, *contact);
    }
}

```

jun 29, 18 16:28	ContactEventListener.cpp	Page 2/2
------------------	---------------------------------	----------

```

    }

    fixtureData = contact->GetFixtureB()->GetUserData();
    if (fixtureData) {
        Worms::PhysicsEntity *sensor = static_cast<Worms::TouchSensor *>(fixtureData);
        sensor->startContact(playerA, *contact);
    }
}

void ContactEventListener::EndContact(b2Contact *contact) {
    Worms::PhysicsEntity *playerA =
        static_cast<Worms::PhysicsEntity *>(contact->GetFixtureA()->GetBody()->GetUserData());
    Worms::PhysicsEntity *playerB =
        static_cast<Worms::PhysicsEntity *>(contact->GetFixtureB()->GetBody()->GetUserData());

    if (playerA) {
        playerA->endContact(playerB);
    }
    if (playerB) {
        playerB->endContact(playerA);
    }

    void *fixtureData = contact->GetFixtureA()->GetUserData();
    if (fixtureData) {
        Worms::PhysicsEntity *sensor = static_cast<Worms::TouchSensor *>(fixtureData);
        sensor->endContact(playerB, *contact);
    }

    fixtureData = contact->GetFixtureB()->GetUserData();
    if (fixtureData) {
        Worms::PhysicsEntity *sensor = static_cast<Worms::TouchSensor *>(fixtureData);
        sensor->endContact(playerA, *contact);
    }
}

```

jun 26, 18 2:39

ContactEventListener.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/05/18
 */

#ifndef __ContactEventListener_H__
#define __ContactEventListener_H__

#include "Box2D/Box2D.h"

class ContactEventListener : public b2ContactListener {
public:
    ContactEventListener() = default;
    ~ContactEventListener() = default;

    void PreSolve(b2Contact* contact, const b2Manifold* oldManifold) override;
    void BeginContact(b2Contact* contact) override;
    void EndContact(b2Contact* contact) override;
};

#endif //__ContactEventListener_H__
```

jun 26, 18 2:39

Dead.cpp

Page 1/1

```
/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#include "Dead.h"
#include "../Player.h"

Worms::Dead::Dead() : State(Worm::StateID::Dead) {}

void Worms::Dead::update(Worms::Player &p, float dt, b2Body *body) {}

void Worms::Dead::moveRight(Worms::Player &p) {}

void Worms::Dead::moveLeft(Worms::Player &p) {}

void Worms::Dead::jump(Worms::Player &p) {}

void Worms::Dead::stopMove(Worms::Player &p) {}

void Worms::Dead::backFlip(Worms::Player &p) {}

void Worms::Dead::bazooka(Worms::Player &p) {}

void Worms::Dead::pointUp(Worms::Player &p) {}

void Worms::Dead::pointDown(Worms::Player &p) {}

void Worms::Dead::startShot(Worms::Player &p) {}

void Worms::Dead::endShot(Worms::Player &p) {}

void Worms::Dead::grenade(Worms::Player &p) {}

void Worms::Dead::cluster(Worms::Player &p) {}

void Worms::Dead::mortar(Worms::Player &p) {}

void Worms::Dead::banana(Worms::Player &p) {}

void Worms::Dead::holy(Worms::Player &p) {}

void Worms::Dead::setTimeout(Worms::Player &p, uint8_t time) {}

void Worms::Dead::aerialAttack(Worms::Player &p) {}

void Worms::Dead::dynamite(Worms::Player &p) {}

void Worms::Dead::teleport(Worms::Player &p) {}

void Worms::Dead::baseballBat(Worms::Player &p) {}
```

jun 26, 18 2:39

Dead.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#ifndef __Dead_H__
#define __Dead_H__

#include <cstdint>
#include "PlayerState.h"

namespace Worms {
class Dead : public State {
public:
    Dead();
    ~Dead() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;
};
} // namespace Worms

#endif // __Dead_H__

```


jun 29, 18 16:28

Die.cpp

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#include "Die.h"
#include "../Player.h"

Worms::Die::Die() : State(Worm::StateID::Die) {}

void Worms::Die::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed >= this->dyingTime) {
        if (p.dyingDisconnected) {
            p.notify(p, Event::DeadDueToDisconnection);
        } else {
            p.notify(p, Event::Dead);
        }
        p.setState(Worm::StateID::Dead);
    }
}

void Worms::Die::moveRight(Worms::Player &p) {}
void Worms::Die::moveLeft(Worms::Player &p) {}
void Worms::Die::jump(Worms::Player &p) {}
void Worms::Die::stopMove(Worms::Player &p) {}
void Worms::Die::backFlip(Worms::Player &p) {}
void Worms::Die::bazooka(Worms::Player &p) {}
void Worms::Die::pointUp(Worms::Player &p) {}
void Worms::Die::pointDown(Worms::Player &p) {}
void Worms::Die::startShot(Worms::Player &p) {}
void Worms::Die::endShot(Worms::Player &p) {}
void Worms::Die::grenade(Worms::Player &p) {}
void Worms::Die::cluster(Worms::Player &p) {}
void Worms::Die::mortar(Worms::Player &p) {}
void Worms::Die::banana(Worms::Player &p) {}
void Worms::Die::holy(Worms::Player &p) {}
void Worms::Die::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::Die::aerialAttack(Worms::Player &p) {}
void Worms::Die::dynamite(Worms::Player &p) {}
void Worms::Die::teleport(Worms::Player &p) {}
void Worms::Die::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

Die.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#ifndef __DIE_H__
#define __DIE_H__

#include <stdint>
#include "../Config/Config.h"
#include "PlayerState.h"

namespace Worms {
class Die : public State {
public:
    Die();
    ~Die() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    float dyingTime{Game::Config::getInstance().getDyingTime()};
};
} // namespace Worms

#endif //__DIE_H__

```

jun 26, 18 2:39

Drowning.cpp

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 29/05/18
 */

#include "Drowning.h"
#include "../Player.h"

Worms::Drowning::Drowning()
    : State(Worm::StateID::Drowning), drowningTime(Game::Config::getInstance().getDrowningTime()) {}

void Worms::Drowning::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed >= this->drowningTime) {
        p.setState(Worm::StateID::Dead);
        p.notify(p, Event::Drowned);
    }
}

void Worms::Drowning::moveRight(Worms::Player &p) {}
void Worms::Drowning::moveLeft(Worms::Player &p) {}
void Worms::Drowning::jump(Worms::Player &p) {}
void Worms::Drowning::stopMove(Worms::Player &p) {}
void Worms::Drowning::backFlip(Worms::Player &p) {}
void Worms::Drowning::bazooka(Worms::Player &p) {}
void Worms::Drowning::pointUp(Worms::Player &p) {}
void Worms::Drowning::pointDown(Worms::Player &p) {}
void Worms::Drowning::startShot(Worms::Player &p) {}
void Worms::Drowning::endShot(Worms::Player &p) {}
void Worms::Drowning::grenade(Worms::Player &p) {}
void Worms::Drowning::cluster(Worms::Player &p) {}
void Worms::Drowning::mortar(Worms::Player &p) {}
void Worms::Drowning::banana(Worms::Player &p) {}
void Worms::Drowning::holy(Worms::Player &p) {}
void Worms::Drowning::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::Drowning::aerialAttack(Worms::Player &p) {}
void Worms::Drowning::dynamite(Worms::Player &p) {}
void Worms::Drowning::teleport(Worms::Player &p) {}
void Worms::Drowning::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

Drowning.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 29/05/18
 */

#ifndef __Drown_H__
#define __Drown_H__

#include <stdint>

#include "../Config/Config.h"
#include "PlayerState.h"

namespace Worms {
class Drowning : public State {
public:
    Drowning();
    ~Drowning() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

    float timeElapsed{0.0f};
    float drowningTime;
};
} // namespace Worms

#endif //__Drown_H__

```

jun 26, 18 2:39

Dynamite.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#include "Dynamite.h"
#include "../Player.h"

#define CONFIG Game::Config::getInstance()

Weapon::Dynamite::Dynamite()
    : Worms::Weapon(CONFIG.getDynamiteConfig(), Worms::WeaponID::WDynamite, 0.0)
{}

void Weapon::Dynamite::update(float dt) {}

void Weapon::Dynamite::startShot(Worms::Player *player) {}

void Weapon::Dynamite::endShot() {}

void Weapon::Dynamite::setTimeout(uint8_t time) {
    this->timeLimit = time;
}

std::list<Worms::Bullet> Weapon::Dynamite::onExplode(const Worms::Bullet &mainBullet,
                                                    Worms::Physics &physics) {
    return std::list<Worms::Bullet>();
}

void Weapon::Dynamite::positionSelected(Worms::Player &p, Math::Point<float> point) {}

void Weapon::Dynamite::increaseAngle() {}

void Weapon::Dynamite::decreaseAngle() {}

```

jun 26, 18 2:39

Dynamite.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 16/06/18
 */

#ifndef __TNT_H__
#define __TNT_H__

#include "Weapon.h"

namespace Weapon {
class Dynamite : public Worms::Weapon {
public:
    Dynamite();
    ~Dynamite() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    void onExplode(const Worms::Bullet &mainBullet,
                  Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;
    void increaseAngle() override;
    void decreaseAngle() override;
};
} // namespace Weapon

#endif //__TNT_H__

```

jun 26, 18 2:39

EndBackFlip.cpp

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#include "EndBackFlip.h"
#include "../Config/Config.h"
#include "../Player.h"
#include "PlayerState.h"

Worms::EndBackFlip::EndBackFlip()
    : State(Worm::StateID::EndBackFlip), landTime(Game::Config::getInstance().getLandTime()) {}

void Worms::EndBackFlip::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed > this->landTime) {
        p.setState(Worm::StateID::Still);
    }
}

void Worms::EndBackFlip::moveRight(Worms::Player &p) {}
void Worms::EndBackFlip::moveLeft(Worms::Player &p) {}
void Worms::EndBackFlip::jump(Worms::Player &p) {}
void Worms::EndBackFlip::stopMove(Worms::Player &p) {}
void Worms::EndBackFlip::backFlip(Worms::Player &p) {}
void Worms::EndBackFlip::bazooka(Worms::Player &p) {}
void Worms::EndBackFlip::pointUp(Worms::Player &p) {}
void Worms::EndBackFlip::pointDown(Worms::Player &p) {}
void Worms::EndBackFlip::startShot(Worms::Player &p) {}
void Worms::EndBackFlip::endShot(Worms::Player &p) {}
void Worms::EndBackFlip::grenade(Worms::Player &p) {}
void Worms::EndBackFlip::cluster(Worms::Player &p) {}
void Worms::EndBackFlip::mortar(Worms::Player &p) {}
void Worms::EndBackFlip::banana(Worms::Player &p) {}
void Worms::EndBackFlip::holy(Worms::Player &p) {}
void Worms::EndBackFlip::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::EndBackFlip::aerialAttack(Worms::Player &p) {}
void Worms::EndBackFlip::dynamite(Worms::Player &p) {}
void Worms::EndBackFlip::teleport(Worms::Player &p) {}
void Worms::EndBackFlip::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

EndBackFlip.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 21/05/18
 */

#ifndef __PLAYER_END_BACKFLIP_H__
#define __PLAYER_END_BACKFLIP_H__

#include <cstdint>
#include "PlayerState.h"

namespace Worms {
class EndBackFlip : public State {
public:
    EndBackFlip();
    ~EndBackFlip() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    float landTime;
};
} // namespace Worms

#endif // __PLAYER_END_BACKFLIP_H__

```


jun 26, 18 2:39

EndJump.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/05/18
 */

#include "EndJump.h"
#include "../Config/Config.h"
#include "../Player.h"

Worms::EndJump::EndJump()
: State(Worm::StateID::EndJump), landTime(Game::Config::getInstance().getLandTime()) {}

void Worms::EndJump::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed > this->landTime) {
        p.setState(Worm::StateID::Still);
    }
}

void Worms::EndJump::moveRight(Worms::Player &p) {}
void Worms::EndJump::moveLeft(Worms::Player &p) {}
void Worms::EndJump::jump(Worms::Player &p) {}
void Worms::EndJump::stopMove(Worms::Player &p) {}
void Worms::EndJump::bazooka(Worms::Player &p) {}
void Worms::EndJump::pointUp(Worms::Player &p) {}
void Worms::EndJump::pointDown(Worms::Player &p) {}
void Worms::EndJump::backFlip(Worms::Player &p) {}
void Worms::EndJump::startShot(Worms::Player &p) {}
void Worms::EndJump::endShot(Worms::Player &p) {}
void Worms::EndJump::grenade(Worms::Player &p) {}
void Worms::EndJump::cluster(Worms::Player &p) {}
void Worms::EndJump::mortar(Worms::Player &p) {}
void Worms::EndJump::banana(Worms::Player &p) {}
void Worms::EndJump::holy(Worms::Player &p) {}
void Worms::EndJump::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::EndJump::aerialAttack(Worms::Player &p) {}
void Worms::EndJump::dynamite(Worms::Player &p) {}
void Worms::EndJump::teleport(Worms::Player &p) {}
void Worms::EndJump::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

EndJump.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/05/18
 */

#ifndef __PLAYER_END_JUMP_H__
#define __PLAYER_END_JUMP_H__

#include "PlayerState.h"

namespace Worms {
class EndJump : public State {
public:
    EndJump();
    ~EndJump() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    virtual void pointUp(Player &p) override;
    virtual void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    const float landTime;
};
} // namespace Worms

#endif //__PLAYER_END_JUMP_H__

```

jun 26, 18 2:39

Falling.cpp

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#include "Falling.h"

Worms::Falling::Falling(GUI::Position p) : State(Worm::StateID::Falling), startPosition(p) {}

void Worms::Falling::update(Player &p, float dt, b2Body *body) {
    if (p.isOnGround()) {
        p.landDamage(this->startPosition.y - p.getPosition().y);
        p.setState(Worm::StateID::Land);
    }
}

void Worms::Falling::moveRight(Worms::Player &p) {}
void Worms::Falling::moveLeft(Worms::Player &p) {}
void Worms::Falling::jump(Worms::Player &p) {}
void Worms::Falling::stopMove(Worms::Player &p) {}
void Worms::Falling::backFlip(Worms::Player &p) {}
void Worms::Falling::bazooka(Worms::Player &p) {}
void Worms::Falling::pointUp(Worms::Player &p) {}
void Worms::Falling::pointDown(Worms::Player &p) {}
void Worms::Falling::startShot(Worms::Player &p) {}
void Worms::Falling::endShot(Worms::Player &p) {}
void Worms::Falling::grenade(Worms::Player &p) {}
void Worms::Falling::cluster(Worms::Player &p) {}
void Worms::Falling::mortar(Worms::Player &p) {}
void Worms::Falling::banana(Worms::Player &p) {}
void Worms::Falling::holy(Worms::Player &p) {}
void Worms::Falling::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::Falling::aerialAttack(Worms::Player &p) {}
void Worms::Falling::dynamite(Worms::Player &p) {}
void Worms::Falling::teleport(Worms::Player &p) {}
void Worms::Falling::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

Falling.h

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#ifndef INC_4_WORMS_FALLING_H
#define INC_4_WORMS_FALLING_H

#include <Camera.h>
#include <cstdlib>
#include "../Player.h"

namespace Worms {
class Falling : public State {
public:
    Falling(GUI::Position p);
    ~Falling() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;
    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    GUI::Position startPosition;
};
} // namespace Worms

#endif // INC_4_WORMS_FALLING_H

```

jun 26, 18 2:39

GameClock.cpp

Page 1/1

```

//
// Created by rodrigo on 10/06/18.
//

#include "GameClock.h"

GameClock::GameClock()
: turnTime(Game::Config::getInstance().getTurnTime()),
  extraTurnTime(Game::Config::getInstance().getExtraTurnTime()),
  currentTurnTime(turnTime),
  waitForNextTurnTime(Game::Config::getInstance().getWaitForNextTurnTime())
{}

void GameClock::playerShot() {
    this->currentTurnTime = this->extraTurnTime;
    this->timeElapsed = 0.0f;
}

void GameClock::update(float dt) {
    this->timeElapsed += dt;
    if (this->timeElapsed > this->currentTurnTime) {
        if (this->waitingForNextTurn) {
            this->notify(*this, Event::NextTurn);
        } else {
            this->notify(*this, Event::EndTurn);
        }
    }
}

double GameClock::getTimeElapsed() const {
    return this->timeElapsed;
}

double GameClock::getTurnTime() const {
    return this->currentTurnTime;
}

void GameClock::restart() {
    this->waitingForNextTurn = false;
    this->timeElapsed = 0.0f;
    this->currentTurnTime = this->turnTime;
}

void GameClock::endTurn() {
    this->timeElapsed = this->currentTurnTime + 1.0f;
    this->notify(*this, Event::EndTurn);
}

void GameClock::waitForNextTurn() {
    this->timeElapsed = 0.0f;
    this->currentTurnTime = this->waitForNextTurnTime;
    this->waitingForNextTurn = true;
}

```

jun 26, 18 2:39

GameClock.h

Page 1/1

```
//  
// Created by rodrigo on 10/06/18.  
//  
  
#ifndef INC_4_WORMS_GAMECLOCK_H  
#define INC_4_WORMS_GAMECLOCK_H  
  
#include "Config/Config.h"  
#include "Subject.h"  
  
class GameClock : public Subject {  
public:  
    GameClock();  
    ~GameClock() = default;  
    void update(float dt);  
    void playerShot();  
    double getTimeElapsed() const;  
    double getTurnTime() const;  
    void waitForNextTurn();  
    void restart();  
    void endTurn();  
  
private:  
    float timeElapsed{0.0f};  
    float turnTime;  
    float extraTurnTime;  
    float currentTurnTime;  
    float waitForNextTurnTime;  
    bool waitingForNextTurn{false};  
};  
  
#endif // INC_4_WORMS_GAMECLOCK_H
```

jun 29, 18 16:28 **Game.cpp** Page 1/10

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#include <Stage.h>
#include <zconf.h>
#include <atomic>
#include <cassert>
#include <chrono>
#include <iostream>
#include <random>
#include "Box2D/Box2D.h"
#include "Chronometer.h"

#include "Config/Config.h"
#include "Direction.h"
#include "Game.h"
#include "GameStates/ImpactOnCourse.h"
#include "Player.h"
#include "Stage.h"
#include "Weapons/BaseballBat.h"

#define CONFIG ::Game::Config::getInstance()
#define TIME_STEP (1.0f / 30.0f)

Worms::Game::Game(Stage &stage, std::vector<CommunicationSocket> &sockets)
: physics(b2Vec2{0.0f, -10.0f}, TIME_STEP),
  stage(std::move(stage)),
  maxTurnTime(::Game::Config::getInstance().getExtraTurnTime()),
  gameTurn(*this),
  sockets(sockets),
  inputs(sockets.size()),
  snapshots(sockets.size()),
  playersConnected(sockets.size()) {
    this->inputThreads.reserve(sockets.size());
    this->outputThreads.reserve(sockets.size());
    for (std::size_t i = 0; i < sockets.size(); i++) {
        this->inputThreads.emplace_back([this, i] { this->inputWorker(i); });
        this->outputThreads.emplace_back([this, i] { this->outputWorker(i); });
    }
    /* reserves the required space to avoid reallocations that may move the worm
    addresses */
    this->players.reserve(this->stage.getWorms().size());
    uint8_t id = 0;
    for (auto &wormData : this->stage.getWorms()) {
        /* initializes the instances */
        this->players.emplace_back(this->physics);
        this->players.back().setPosition(wormData.position);
        this->players.back().health = wormData.health;
        this->players.back().setId(id);
        this->players.back().addObserver(this);
        id++;
    }

    this->teams.makeTeams(this->players, (uint8_t)sockets.size(), this->stage.ge
tAmmoCounter());
    // this->wind.range = CONFIG.getWindIntensityRange();
    this->wind.minIntensity = CONFIG.getMinWindIntensity();
    this->wind.maxIntensity = CONFIG.getMaxWindIntensity();
    this->calculateWind();

```

jun 29, 18 16:28 **Game.cpp** Page 2/10

```

/* sets the girders */
this->girders.reserve(this->stage.getGirders().size());
for (auto &girder : this->stage.getGirders()) {
    this->girders.emplace_back(girder, this->physics);
}

/* calculate the initial team's healths */
this->teamHealths = this->teams.getTotalHealth(this->players);

this->currentWorm = this->teams.getCurrentPlayerID();
this->currentWormToFollow = this->currentWorm;

this->gameClock.addObserver(this);
this->gameClock.waitForNextTurn();
}

Worms::Game::~Game() {
    this->exit();
    for (auto &t : this->outputThreads) {
        t.join();
    }

    for (auto &t : this->inputThreads) {
        t.join();
    }
}

/**
 * @brief Reads player messages from a socket and pushes them into the input q
ueue.
 *
 * @param playerIndex The index of the player.
 */
void Worms::Game::inputWorker(std::size_t playerIndex) {
    PlayerInput &input = this->inputs.at(playerIndex);
    CommunicationSocket &socket = this->sockets.at(playerIndex);

    /* TODO: avoid hardcoding the size */
    IO::PlayerMsg msg;
    char *buffer = new char[msg.getSerializedSize()];

    try {
        while (!this->quit) {
            /* reads the raw data from the buffer */
            socket.receive(buffer, msg.getSerializedSize());

            /* sets the struct data from the buffer */
            msg.deserialize(buffer, msg.getSerializedSize());

            /* pushes the message into the player's input queue if it's the cu
rrent player */
            if (this->currentTeam == playerIndex) {
                input.push(msg);
            }
        }
    } catch (const std::exception &e) {
        std::cerr << "Worms::Game::inputWorker:" << e.what() << std::endl;
        msg.input = IO::PlayerInput::disconnected;
        msg.position = Math::Point<float>{0, 0};
        input.push(msg);
    } catch (...) {
        std::cerr << "Unknown error in Worms::Game::inputWorker()" << std::end
l;

```

jun 29, 18 16:28	Game.cpp	Page 3/10
------------------	-----------------	-----------

```

//      }
//
//      delete[] buffer;
//}
//
///**
// * @brief Sends model snapshot messages to a socket.
// *
// * @param playerIndex The index of the player to send the spanshots to.
// */
//void Worms::Game::outputWorker(std::size_t playerIndex) {
//    CommunicationSocket &socket = this->sockets.at(playerIndex);
//    GameSnapshot &snapshot = this->snapshots.at(playerIndex);
//
//    IO::GameStateMsg msg;
//    char *buffer = new char[msg.getSerializedSize()];
//
//    try {
//        while (!this->quit) {
//            msg = snapshot.get(true);
//            msg.serialize(buffer, msg.getSerializedSize());
//            socket.send(buffer, msg.getSerializedSize());
//        }
//    } catch (const IO::Interrupted &e) {
//        /* this means that the game is ready to exit */
//    } catch (const std::exception &e) {
//        std::cerr << "Worms::Game::outputWorker:" << e.what() << std::endl;
//    } catch (...) {
//        std::cerr << "Unknown error in Worms::Game::outputWorker()" << std::en
dl;
//    }
//
//    delete[] buffer;
//}

/**
 * @brief Reads player messages from a socket and pushes them into the input que
ue.
 *
 * @param playerIndex The index of the player.
 */
void Worms::Game::inputWorker(std::size_t playerIndex) {
    PlayerInput &input = this->inputs.at(playerIndex);
    CommunicationSocket &socket = this->sockets.at(playerIndex);

    /* TODO: avoid hardcoding the size */
    IO::PlayerMsg msg;
    try {
        while (!this->quit) {
            /* receives the size of the msg */
            std::uint32_t size(0);
            socket.receive((char *)&size, sizeof(std::uint32_t));
            size = ntohl(size);

            std::vector<char> buffer(size, 0);
            /* reads the raw data from the buffer */
            socket.receive(buffer.data(), size);

            std::string buff(buffer.data(), size);

            /* sets the struct data from the buffer */

```

jun 29, 18 16:28	Game.cpp	Page 4/10
------------------	-----------------	-----------

```

        msg.deserialize(buff);

        /* pushes the message into the player's input queue if it's the curr
ent player */
        if (this->currentTeam == playerIndex) {
            input.push(msg);
        }
    } catch (const std::exception &e) {
        std::cerr << "Worms::Game::inputWorker:" << e.what() << std::endl;
        msg.input = IO::PlayerInput::disconnected;
        msg.position = Math::Point<float>(0, 0);
        input.push(msg);
    } catch (...) {
        std::cerr << "Unknown error in Worms::Game::inputWorker()" << std::endl;
    }
}

/**
 * @brief Sends model snapshot messages to a socket.
 *
 * @param playerIndex The index of the player to send the spanshots to.
 */
void Worms::Game::outputWorker(std::size_t playerIndex) {
    CommunicationSocket &socket = this->sockets.at(playerIndex);
    GameSnapshot &snapshot = this->snapshots.at(playerIndex);

    IO::GameStateMsg msg;
    try {
        while (!this->quit) {
            msg = snapshot.get(true);
            std::string buff = msg.serialize();
            std::uint32_t size = buff.size();
            std::uint32_t netInt = htonl(size);

            socket.send((char *)&netInt, sizeof(std::uint32_t));
            socket.send(buff.data(), size);
        }
    } catch (const IO::Interrupted &e) {
        /* this means that the game is ready to exit */
    } catch (const std::exception &e) {
        std::cerr << "Worms::Game::outputWorker:" << e.what() << std::endl;
    }
}

void Worms::Game::start() {
    try {
        /* game loop */
        Utils::Chronometer chronometer;
        while (!quit) {
            double dt = chronometer.elapsed();

            this->gameClock.update(dt);
            this->gameTurn.update(dt);

            IO::PlayerMsg pMsg;
            if (this->inputs.at(this->currentTeam).pop(pMsg, false)) {
                if (pMsg.input == IO::PlayerInput::disconnected) {
                    this->playerDisconnected(this->currentTeam);
                } else {
                    if (this->processingClientInputs) {
                        if (this->currentPlayerShot) {

```


jun 29, 18 16:28

Game.cpp

Page 5/10

```

        if (pMsg.input != IO::PlayerInput::startShot &&
            pMsg.input != IO::PlayerInput::endShot &&
            pMsg.input != IO::PlayerInput::positionSelected)
        {
            this->players.at(this->currentWorm).handleState(
pMsg);
        }
        else {
            this->players.at(this->currentWorm).handleState(pMsg);
        }
        else {
            this->players.at(this->currentWorm).handleState(pMsg);
        }
    }

    /* updates the actors */
    for (auto &worm : this->players) {
        worm.update(dt);
    }

    for (auto &bullet : this->bullets) {
        bullet.update(dt, this->wind);
    }

    this->physics.update(dt);

    /* serializes and updates the game state */
    auto msg = this->serialize();
    for (auto &snapshot : this->snapshots) {
        snapshot.set(msg);
        snapshot.swap();
    }

    if (this->gameEnded) {
        this->quit = true;
    }

    if (TIME_STEP > dt) {
        usleep((TIME_STEP - dt) * 1000000);
    }
}

} catch (std::exception &e) {
    std::cerr << e.what() << std::endl << "In Worms::Game::start" << std::endl;
} catch (...) {
    std::cerr << "Unkown error in Worms::Game::start()" << std::endl;
}

}

void Worms::Game::endTurn() {
    this->waitingForNextTurn = false;
    this->processingClientInputs = true;
    this->gameClock.restart();
    this->gameTurn.restart();
    this->calculateWind();
}

void Worms::Game::calculateCurrentPlayer() {
    this->waitingForNextTurn = true;
    this->players[this->currentWorm].reset();
    this->gameEnded = this->teams.endTurn(this->players);
}

```

jun 29, 18 16:28

Game.cpp

Page 6/10

```

    if (this->gameEnded) {
        this->winnerTeam = this->teams.getWinner();
    }
    this->currentTeam = this->teams.getCurrentTeamID();
    this->currentWorm = this->teams.getCurrentPlayerID();
    this->currentWormToFollow = this->currentWorm;
}

IO::GameStateMsg Worms::Game::serialize() const {
    assert(this->players.size() <= 20);

    IO::GameStateMsg m;
    memset(&m, 0, sizeof(m));

    m.num_worms = 0;
    m.num_teams = this->teams.getTeamQuantity();
    for (const auto &worm : this->players) {
        m.positions[m.num_worms * 2] = worm.getPosition().x;
        m.positions[m.num_worms * 2 + 1] = worm.getPosition().y;
        m.stateIDs[m.num_worms] = worm.getStateId();
        m.wormsHealth[m.num_worms] = worm.health;
        m.wormsTeam[m.num_worms] = worm.getTeam();
        m.wormsDirection[m.num_worms] = worm.direction;
        m.num_worms++;
    }

    /* sets team health*/
    uint8_t i{0};
    for (auto health : this->teamHealths) {
        m.teamHealths[i++] = health;
    }

    /* sets wind data */
    m.windIntensity =
        (char)(127.0f * this->wind.instensity /
            (this->wind.maxIntensity - this->wind.minIntensity) * this->wind.
xDirection);

    /* sets the current player's data */
    m.elapsedTurnSeconds = static_cast<std::uint16_t>(std::floor(this->gameClock
.getTimeElapsed()));
    m.currentPlayerTurnTime = static_cast<std::uint16_t>(std::floor(this->gameCl
ock.getTurnTime()));
    m.currentWorm = this->currentWorm;
    m.currentWormToFollow = this->currentWormToFollow;
    m.currentTeam = this->currentTeam;
    m.activePlayerAngle = this->players[this->currentWorm].getWeaponAngle();
    m.activePlayerWeapon = this->players[this->currentWorm].getWeaponID();

    m.bulletsQuantity = this->bullets.size();
    i = 0;
    uint8_t j = 0;
    for (auto &bullet : this->bullets) {
        Math::Point<float> p = bullet.getPosition();
        m.bullets[i++] = p.x;
        m.bullets[i++] = p.y;
        m.bulletsAngle[j] = bullet.getAngle();
        m.bulletType[j++] = bullet.getWeaponID();
    }
    /*
     * serialize the ammunition counter
     */
    this->teams.serialize(m);
}

```

jun 29, 18 16:28	Game.cpp	Page 7/10
	<pre> m.processingInputs = this->processingClientInputs; m.playerUsedTool = this->currentPlayerShot; m.waitingForNextTurn = this->waitingForNextTurn; m.gameEnded = this->gameEnded; m.winner = this->winnerTeam; return m; } void Worms::Game::exit() { this->quit = true; for (auto &snapshot : this->snapshots) { snapshot.interrupt(); } for (auto &socket : this->sockets) { socket.shutdown(); } } void Worms::Game::onNotify(Subject &subject, Event event) { switch (event) { /** * Because i didnt want to move all responsibility of the bullets to * the game (until the refactor of the start), i added this function * that delegates to the player the responsibility to iterate all over * the bullets and add the game as an observer */ case Event::Shot: { // this->players[this->currentWorm].addObserverToBullets (this); this->bullets.merge(this->players[this->currentWorm].getBullets()); for (auto &bullet : this->bullets) { bullet.addObserver(this); } this->gameClock.playerShot(); this->gameTurn.playerShot(this->players[this->currentWorm].getWeapon ID()); this->currentPlayerShot = true; break; } /** * On explode, the game must check worms health. */ case Event::Explode: { auto &bullet = dynamic_cast<const Bullet &>(subject); this->gameTurn.explosion(); this->calculateDamage(bullet); break; } case Event::P2PWeaponUsed: { auto &player = dynamic_cast<const Worms::Player &>(subject); const std::shared_ptr<Worms::Weapon> weapon = player.getWeapon(); this->gameClock.playerShot(); this->gameTurn.playerShot(this->players[this->currentWorm].getWeapon ID()); this->currentPlayerShot = true; this->gameTurn.explosion(); this->calculateDamage(weapon, player.getPosition(), player.direction); break; } } /** </pre>	

jun 29, 18 16:28	Game.cpp	Page 8/10
	<pre> * onExplode will create new Bullets in player's container, and we * need to listen to them. */ case Event::OnExplode: { auto &bullet = dynamic_cast<const Bullet &>(subject); this->calculateDamage(bullet); this->bullets.merge(this->players[this->currentWorm].onExplode(bulle t, this->physics)); for (auto &fragment : this->bullets) { fragment.addObserver(this); } // this->players[this->currentWorm].addObserverToBullets(this); break; } case Event::DyingDueToDisconnection: { this->gameTurn.playerDisconnected(dynamic_cast<const Player &>(subje ct).getId()); break; } case Event::DeadDueToDisconnection: { this->gameTurn.playerDisconnectedDead(dynamic_cast<const Player &>(s ubject).getId()); break; } case Event::Teleported: { this->gameClock.playerShot(); this->currentPlayerShot = true; this->teams.weaponUsed(this->players[this->currentWorm].getWeaponID()); break; } case Event::WormFalling: { this->gameTurn.wormFalling(dynamic_cast<const Player &>(subject).get Id()); break; } case Event::WormLanded: { this->gameTurn.wormLanded(dynamic_cast<const Player &>(subject).getI d()); break; } case Event::Hit: { this->gameTurn.wormHit(dynamic_cast<const Player &>(subject).getId()); break; } case Event::EndHit: { this->gameTurn.wormEndHit(dynamic_cast<const Player &>(subject).getI d()); break; } case Event::Drowning: { this->gameTurn.wormDrowning(dynamic_cast<const Player &>(subject).ge tId()); break; } case Event::Drowned: { this->gameTurn.wormDrowned(dynamic_cast<const Player &>(subject).get Id()); </pre>	

jun 29, 18 16:28	Game.cpp	Page 9/10
	<pre> break; } case Event::Dying: { this->gameTurn.wormDying(); break; } case Event::Dead: { this->gameTurn.wormDead(); this->gameTurn.endTurn(); break; } case Event::NewWormToFollow: { this->currentWormToFollow = dynamic_cast<const GameTurnState &>(subject).getWormToFollow(); break; } case Event::DamageOnLanding: { this->gameClock.endTurn(); break; } case Event::ImpactEnd: { auto &wormsHit = dynamic_cast<ImpactOnCourse &>(subject).getWormsHit ()); for (auto worm : wormsHit) { Worm::StateID wormState = this->players[worm].getStateId(); if (this->players[worm].health == 0) { if (wormState != Worm::StateID::Die && wormState != Worm::St ateID::Dead) { this->players[worm].notify(this->players[worm], Event::D ying); this->players[worm].setState(Worm::StateID::Die); } } } break; } case Event::EndTurn: { this->processingClientInputs = false; this->gameTurn.endTurn(); break; } case Event::TurnEnded: { if (this->players[this->currentWorm].getStateId() != Worm::StateID:: Dead) { this->players[this->currentWorm].setState(Worm::StateID::Still); } this->bullets.erase(this->bullets.begin(), this->bullets.end()); this->gameClock.waitForNextTurn(); this->teamHealths = this->teams.getTotalHealth(this->players); this->calculateCurrentPlayer(); break; } case Event::NextTurn: { this->currentPlayerShot = false; this->endTurn(); break; } default: { break; } } } </pre>	

jun 29, 18 16:28	Game.cpp	Page 10/10
	<pre> void Worms::Game::calculateDamage(const Worms::Bullet &bullet) { Config::Bullet::DamageInfo damageInfo = bullet.getDamageInfo(); for (auto &worm : this->players) { worm.acknowledgeDamage(damageInfo, bullet.getPosition()); } this->removeBullets = true; } /** * @brief calculate damage for p2p weapons. Because the only one is the * baseball bat and because we are running out of time, there will be * a cast to a baseballWeapon. * TODO make a class between weapon and baseballBat, that represents a * p2pWeapon. * @param weapon */ void Worms::Game::calculateDamage(std::shared_ptr<Worms::Weapon> weapon, Math::Point<float> shooterPosition, Worm::Direction shooterDirection) { auto *baseball = (::Weapon::BaseballBat *)weapon.get(); Config::P2PWeapon &weaponInfo = baseball->getWeaponInfo(); for (auto &worm : this->players) { worm.acknowledgeDamage(weaponInfo, shooterPosition, shooterDirection); } this->removeBullets = true; } void Worms::Game::calculateWind() { std::random_device rnd_device; std::mt19937 mersenne_engine(rnd_device()); std::uniform_real_distribution<> distr(this->wind.minIntensity, this->wi nd.maxIntensity); this->wind.xDirection = (distr(mersenne_engine) > (this->wind.maxIntensity - this->wind.minI ntensity) / 2.0f) ? 1 : -1; this->wind.instensity = (float) distr(mersenne_engine); } void Worms::Game::playerDisconnected(uint8_t teamDisconnected) { this->playersConnected--; this->teams.kill(teamDisconnected, this->players); if (this->playersConnected <= 1) { this->winnerTeam = this->teams.getWinner(); this->gameEnded = true; } } </pre>	

jun 29, 18 16:28	Game.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #ifndef __GAME_H__ #define __GAME_H__ #include <atomic> #include <list> #include <thread> #include <unordered_map> #include "CommunicationSocket.h" #include "Direction.h" #include "DoubleBuffer.h" #include "GameClock.h" #include "GameTeams.h" #include "GameTurn.h" #include "Girder.h" #include "Observer.h" #include "Player.h" #include "Stage.h" #include "Weapons/Bullet.h" namespace Worms { using PlayerInput = IO::Stream<IO::PlayerMsg>; using GameSnapshot = IO::DoubleBuffer<IO::GameStateMsg>; struct Teamasd { std::vector<uint8_t> players; uint8_t currentPlayer; bool alive; }; class Game : Observer { public: std::atomic<bool> quit{false}; Game(Stage &&stage, std::vector<CommunicationSocket> &sockets); virtual ~Game(); Game(Game &&other) = delete; void start(); IO::GameStateMsg serialize() const; void onNotify(Subject &subject, Event event) override; /** * @brief calculates damage for weapons that throw bullets. It gives * information of the bullet to all players so them can calculate his damage * and apply an impulse if this was hit. * @param bullet */ void calculateDamage(const Bullet &bullet); /** * @brief calculates damage for p2p weapons (baseball). It gives * information of the weapon (direction, point and damageInfo) to the * players so that they can calculate his damage and apply an impulse if * this was hit. * @param weapon */ void calculateDamage(std::shared_ptr<Worms::Weapon> weapon, Math::Point<float> </pre>		

jun 29, 18 16:28	Game.h	Page 2/2
<pre> t> shooterPosition, Worm::Direction shooterDirection); void calculateWind(); void exit(); void endTurn(); private: void inputWorker(std::size_t playerIndex); void outputWorker(std::size_t playerIndex); void calculateCurrentPlayer(); uint8_t currentWorm; uint8_t currentTeam{0}; Physics physics; Stage stage; std::vector<Girder> girders; std::vector<Player> players; std::vector<std::uint32_t> teamHealths; const double maxTurnTime; bool processingClientInputs{false}; uint8_t currentWormToFollow{0}; bool currentPlayerShot{false}; GameTeams teams; std::list<Bullet> bullets; Config::Wind wind; std::vector<uint8_t> deadTeams; GameClock gameClock; GameTurn gameTurn; /* communication */ std::vector<std::thread> inputThreads; std::vector<std::thread> outputThreads; std::vector<CommunicationSocket> &sockets; std::vector<PlayerInput> inputs; std::vector<GameSnapshot> snapshots; std::uint8_t playersConnected; bool removeBullets{false}; bool gameEnded{false}; std::uint8_t winnerTeam{0}; bool waitingForNextTurn{true}; void playerDisconnected(uint8_t teamDisconnected); }; } // namespace Worms #endif //__GAME_H__ </pre>		

```

jun 29, 18 16:28      GameLobbyAssistant.cpp      Page 1/3
//
// Created by rodrigo on 15/06/18.
//

#include <fstream>
#include <iostream>

#include "GameLobbyAssistant.h"
#include <GameStateMsg.h>
#include "Protocol.h"
#include "Lobbies.h"
#include "GamesGetter.h"

Worms::GameLobbyAssistant::GameLobbyAssistant(CommunicationSocket &&communicationSocket, Lobbies &lobbies, int id,
                                              Observer *lobbyObs) :
    protocol(communicationSocket),
    lobbies(lobbies),
    playerID(id) {
    this->lobbyObservers.emplace_back(lobbyObs);
    this->lobbyObservers.emplace_back(this);
}

void Worms::GameLobbyAssistant::run() {
    try {
        std::uint8_t command{COMMAND_GET_LEVELS};
        while (!this->quit) {
            this->protocol >> command;
            switch (command) {
                case COMMAND_GET_LEVELS:
                    this->getLevels();
                    break;
                case COMMAND_CREATE_GAME:
                    this->createGame();
                    break;
                case COMMAND_GET_GAMES:
                    this->getGames();
                    break;
                case COMMAND_JOIN_GAME:
                    this->joinGame();
                    break;
            }
        }
        //      this->createGame();
        //      this->createGame();
        //      this->createGame();
        //      this->getGames();
    } catch (std::exception &e) {
        std::cerr << "In GameLobbyAssistant::run()" << std::endl;
        std::cerr << e.what() << std::endl;
    } catch (...) {
        std::cerr << "Unkown error in GameLobbyAssistant::run()" << std::endl;
    }
}

void Worms::GameLobbyAssistant::stop() {
    this->finished = true;
    this->protocol.stopCommunication();
}

void Worms::GameLobbyAssistant::getLevels() {
    //      std::vector<IO::LevelInfo> levelsInfo;

```

```

jun 29, 18 16:28      GameLobbyAssistant.cpp      Page 2/3
//      IO::LevelInfo levelInfo{"First Stage", 2};
//      levelsInfo.emplace_back(levelInfo);
//      levelInfo = {"Second Stage", 3};
//      levelsInfo.emplace_back(levelInfo);
//      levelInfo = {"Third Stage", 4};
//      levelsInfo.emplace_back(levelInfo);

    this->protocol << this->lobbies.getLevels();
}

void Worms::GameLobbyAssistant::createGame() {
    uint8_t levelSelected{0};
    this->protocol >> levelSelected;
    this->sendLevelFiles(levelSelected);

    this->lobbies.createGame(this->playerID, this->lobbyObservers, levelSelected);
};
    this->quit = true;
}

void Worms::GameLobbyAssistant::getGames() {
    GamesGetter getter;
    this->lobbies.getGames(getter);
    this->protocol << getter.gamesInfo;
}

void Worms::GameLobbyAssistant::joinGame() {
    std::uint8_t gameID{0};
    std::uint8_t levelID{0};
    this->protocol >> gameID;
    this->protocol >> levelID;
    this->sendLevelFiles(levelID);
    this->lobbies.joinGame(gameID, this->playerID, this);
    this->quit = true;
}

void Worms::GameLobbyAssistant::onNotify(Subject &subject, Event event) {
    switch (event) {
        case Event::NewPlayer: {
            auto &lobby = dynamic_cast<Lobby &>(subject);
            this->protocol << lobby.getActualPlayers();
            break;
        }
        default: {
            break;
        }
    }
}

CommunicationSocket Worms::GameLobbyAssistant::getSocket() {
    return std::move(this->protocol.getSocket());
}

int Worms::GameLobbyAssistant::getPlayerID() const {
    return this->playerID;
}

bool Worms::GameLobbyAssistant::itsOver() const {
    return this->finished;
}

```

jun 29, 18 16:28

GameLobbyAssistant.cpp

Page 3/3

```
void Worms::GameLobbyAssistant::sendLevelFiles(uint8_t level) {
    const IO::LevelData &levelData = this->lobbies.getLevelData(level);
    this->protocol << levelData.levelName;
    std::ifstream levelFile(levelData.levelPath, std::ifstream::binary);
    this->protocol << levelFile;

    this->protocol << levelData.backgroundName;
    for (auto &background : levelData.backgroundPath) {
        std::ifstream backgroundFile(background, std::ifstream::binary);
        if (!backgroundFile) {
        }
        this->protocol << backgroundFile;
    }
}
```

jun 29, 18 16:28

GameLobbyAssistant.h

Page 1/1

```

//
// Created by rodrigo on 15/06/18.
//

#ifndef INC_4_WORMS_GAMELOBBYASSISTANT_H
#define INC_4_WORMS_GAMELOBBYASSISTANT_H

#include <Protocol.h>
#include <sstream>

#include "Thread.h"
#include "Lobbies.h"
#include "Observer.h"

namespace Worms {
    class GameLobbyAssistant : public Thread, public Observer {
    public:
        explicit GameLobbyAssistant(CommunicationSocket &&communicationSocket, L
obbies &lobbies, int id,
                                Observer *lobbyObs);
        GameLobbyAssistant(GameLobbyAssistant &copy) = delete;
        void run() override;
        void stop() override;
        bool itsOver() const;
        void onNotify(Subject &subject, Event event) override;
        int getPlayerID() const;
        CommunicationSocket getSocket();

    private:
        Protocol<CommunicationSocket> protocol;
        Lobbies &lobbies;
        int playerID;
        std::vector<Observer *> lobbyObservers;
        bool finished{false};

        void getLevels();
        void getGames();
        void joinGame();

        void createGame();

        bool quit{false};

        void sendLevelFiles(uint8_t level);
    };
}

#endif //INC_4_WORMS_GAMELOBBYASSISTANT_H

```

jun 29, 18 16:28

GameLobby.cpp

Page 1/4

```
//
// Created by rodrigo on 15/06/18.
//

#include <iostream>
#include <dirent.h>

#include "GameLobby.h"
#include "ServerSocket.h"
#include "Lobbies.h"
#include "Game.h"
#include "LobbyJoiner.h"
#include "Stage.h"

Worms::GameLobby::GameLobby(std::string port) :
    serverSocket(port.c_str()) {
    std::cout << "Se bindeo" << std::endl;
}

void Worms::GameLobby::run() {
    std::string path(RESOURCE_PATH);
    std::vector<IO::LevelData> levels;

    Lobbies lobbies{levels};

    LobbyJoiner lobbyJoiner{lobbies, this->msgToJoiner};
    try {
        this->loadLevels(path, levels);
        lobbies.configure();
        lobbyJoiner.start();
        int id = 0;

        while (!quit) {
            this->players.emplace_back(this->serverSocket.accept(), lobbies, id,
this);
            this->players.back().start();
            id++;

            this->removePlayers();

            std::cout << "hubo una conexiÃ³n" << std::endl;
        }
    } catch (std::exception &e) {
        if (!this->quit) {
            std::cerr << "In GameLobby::run()" << std::endl;
            std::cerr << e.what() << std::endl;
        }
    } catch (...) {
        std::cerr << "Unkown error in GameLobby::run()" << std::endl;
    }

    this->killPlayers();
    this->msgToJoiner << IO::ServerInternalMsg{IO::ServerInternalAction::quit};
    lobbyJoiner.join();
}

void Worms::GameLobby::stop() {
    this->quit = true;
    this->serverSocket.shutdown();
}
```

jun 29, 18 16:28

GameLobby.cpp

Page 2/4

```
void Worms::GameLobby::onNotify(Subject &subject, Event event) {
    switch (event) {
        case Event::StartGame: {
            auto &lobby = dynamic_cast<Lobby &>(subject);

            /** En algÃºn momento le tengo que sacar el socket al GameLobbyAssis
tant
            * para crear un vector con los sockets de todos los jugadores, que
es lo que
            * recibe Game, entonces pienso que es mejor que sea al momento de
iniciar la partida
            * por si el jugador se arrepiente antes y quiere salir, que el Ass
istant lo pueda
            * manejar.
            */

            const std::vector<int> &playerIDs = lobby.getPlayerIDs();
            for (auto &playerID : playerIDs) {
                for (auto &player : this->players) {
                    if (player.getPlayerID() == playerID) {
                        //TODO revisar el lugar donde se setea terminado el hilo
                        lobby.addPlayerSocket(std::move(player.getSocket()));
                        player.stop();
                    }
                }
            }
            lobby.start();

            break;
        }
        case Event::EndGame: {
            this->msgToJoiner << IO::ServerInternalMsg{IO::ServerInternalAction:
:lobbyFinished};
            break;
        }
        default: {
            break;
        }
    }
}

void Worms::GameLobby::removePlayers() {
    std::list<GameLobbyAssistant>::iterator playerIt;
    playerIt = this->players.begin();
    while (playerIt != this->players.end()) {
        if (playerIt->itsOver()) {
            playerIt->join();
            playerIt = this->players.erase(playerIt);
        } else {
            playerIt++;
        }
    }
}

void Worms::GameLobby::loadLevels(std::string &path, std::vector<IO::LevelData>
&levels) {
    DIR *dir;
    struct dirent *ent;
    if ((dir = opendir(path.c_str())) != NULL) {
        /* print all the files and directories within directory */
        while ((ent = readdir(dir)) != NULL) {
            if (std::string(ent->d_name)[0] != '.') {

```


jun 29, 18 16:28

GameLobby.cpp

Page 3/4

```

        std::string levelPath(path + ent->d_name + "/");
        this->loadLevel(levelPath, levels);
    }
    closedir (dir);
} else {
    /* could not open directory */
    throw Exception("Could not open directory: %s", path.c_str());
}

void Worms::GameLobby::loadLevel(std::string &path, std::vector<IO::LevelData> &
levels) {
    DIR *dir;
    struct dirent *ent;
    IO::LevelData level;
    if ((dir = opendir(path.c_str())) != NULL) {
        /* print all the files and directories within directory */
        while ((ent = readdir(dir)) != NULL) {
            if (std::string(ent->d_name)[0] != '.') {
                std::string levelPath(path + ent->d_name);
                if (std::string(ent->d_name) == "Background") {
                    std::string backgroundsPath(levelPath + "/");
                    this->loadLevelBackground(backgroundsPath, level);
                } else {
                    std::string levelName(ent->d_name);
                    YAML::Node data = YAML::LoadFile(levelPath);
                    std::set<char> delims{'/'};
                    std::string closeBackgroundFile = data["background"]["closeBackground
File"].as<std::string>();
                    level.backgroundName.emplace_back(std::move(this->splitpath(clos
eBackgroundFile, delims)));
                    level.backgroundPath.emplace_back(std::move(closeBackgroundFile)
);
                    std::string midBackgroundFile = data["background"]["midBackgroundFile
"].as<std::string>();
                    level.backgroundName.emplace_back(std::move(this->splitpath(midB
ackgroundFile, delims)));
                    level.backgroundPath.emplace_back(std::move(midBackgroundFile));
                    std::string fartherBackgroundFile = data["background"]["fartherBackgro
undFile"].as<std::string>();
                    level.backgroundName.emplace_back(std::move(this->splitpath(fart
herBackgroundFile, delims)));
                    level.backgroundPath.emplace_back(std::move(fartherBackgroundFil
e));

                    levelName = levelName.substr(0, levelName.find('.'));

                    level.levelPath = std::move(levelPath);
                    level.levelName = std::move(levelName);
                }
            }
        }
        closedir (dir);
        levels.emplace_back(std::move(level));
    } else {
        /* could not open directory */
        throw Exception("Could not open directory: %s", path.c_str());
    }
}

```

jun 29, 18 16:28

GameLobby.cpp

Page 4/4

```

std::string Worms::GameLobby::splitpath(const std::string &str, const std::set<c
har> &delimiters) {
    std::vector<std::string> result;

    char const* pch = str.c_str();
    char const* start = pch;
    for(; *pch; ++pch) {
        if (delimiters.find(*pch) != delimiters.end()) {
            if (start != pch) {
                std::string str(start, pch);
                result.push_back(str);
            } else {
                result.emplace_back("");
            }
            start = pch + 1;
        }
    }
    result.emplace_back(start);

    return result.back();
}

void Worms::GameLobby::loadLevelBackground(std::string &path, IO::LevelData &lev
el) {
    DIR *dir;
    struct dirent *ent;
    std::vector<std::string> backgrounds;
    if ((dir = opendir(path.c_str())) != NULL) {
        /* print all the files and directories within directory */
        while ((ent = readdir(dir)) != NULL) {
            if (std::string(ent->d_name)[0] != '.') {
                std::string backgroundPath(path + ent->d_name);
                std::string backgroundName(ent->d_name);

                level.backgroundPath.emplace_back(std::move(backgroundPath));
                level.backgroundName.emplace_back(std::move(backgroundName));
            }
        }
        closedir (dir);
    } else {
        /* could not open directory */
        throw Exception("Could not open directory: %s", path.c_str());
    }
}

void Worms::GameLobby::killPlayers(){
    std::list<GameLobbyAssistant>::iterator playerIt;
    playerIt = this->players.begin();
    while (playerIt != this->players.end()) {
        playerIt->stop();
        playerIt->join();
        playerIt++;
    }
    this->players.erase(this->players.begin(), this->players.end());
}

```

jun 29, 18 16:28

GameLobby.h

Page 1/1

```

//
// Created by rodrigo on 15/06/18.
//

#ifndef INC_4_WORMS_GAMELOBBY_H
#define INC_4_WORMS_GAMELOBBY_H

#define RESOURCE_PATH "/var/Worms/res/"

#include <list>
#include <string>

#include <CommunicationSocket.h>
#include <thread>
#include <GameStateMsg.h>
#include <Stream.h>
#include "ServerSocket.h"
#include "GameLobbyAssistant.h"

namespace Worms {
    class GameLobby : public Observer, public Thread {
    public:
        GameLobby(std::string port);
        GameLobby(GameLobby &copy) = delete;

        void run() override;
        void onNotify(Subject &subject, Event event) override;
        void stop() override;

    private:
        ServerSocket serverSocket;
        IO::Stream<IO::ServerInternalMsg> msgToJoiner;
        std::list<GameLobbyAssistant> players;
        bool quit{false};
        /**
         * @brief check if the GameLobbyAssistant thread is over. If so, join
         * it and erase it (because the sockets was already moved to the Lobby)
         */
        void removePlayers();

        void loadLevels(std::string &path, std::vector<IO::LevelData> &levels);
        void loadLevel(std::string &path, std::vector<IO::LevelData> &levels);
        std::string splitpath(const std::string &str, const std::set<char> &delimiters);
        void loadLevelBackground(std::string &path, IO::LevelData &level);

        void killPlayers();
    };
}

#endif //INC_4_WORMS_GAMELOBBY_H

```

jun 29, 18 16:28

GamesGetter.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 17/06/18
 */

#include "GamesGetter.h"

void GamesGetter::operator()(const std::list<Worms::Lobby> &lobbies){
    for (auto &lobby : lobbies) {
        auto &levelInfo = lobby.getLevelInfo();
        IO::GameInfo gameInfo{lobby.getID(),
                               levelInfo.id,
                               levelInfo.name,
                               lobby.getActualPlayers(),
                               levelInfo.playersQuantity};
        this->gamesInfo.emplace_back(gameInfo);
    }
}
```

jun 29, 18 16:28

GamesGetter.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 17/06/18
 */

#ifndef __GamesGetter_H__
#define __GamesGetter_H__

#include <list>
#include <string>

#include "GameStateMsg.h"
#include "Lobby.h"

struct GamesGetter{
public:
    void operator() (const std::list<Worms::Lobby> &lobbies);
    std::vector<IO::GameInfo> gamesInfo;
};

#endif //__GamesGetter_H__
```

jun 29, 18 16:28	GameTeams.cpp	Page 1/3
------------------	----------------------	----------

```

//
// Created by rodrigo on 3/06/18.
//

#include "GameTeams.h"
#include <random>

void Worms::GameTeams::makeTeams(std::vector<Worms::Player> &players, uint8_t numTeams,
                                const std::map<Worm::WeaponID, std::int16_t> &ammoCounter) {
    uint8_t numPlayers = players.size();

    this->teams.reserve(numTeams);
    std::vector<uint8_t> playersNum(numPlayers);
    for (uint8_t i = 0; i < numPlayers; i++) {
        playersNum[i] = i;
    }

    std::random_device rnd_device;
    std::mt19937 mersenne_engine(rnd_device());

    shuffle(playersNum.begin(), playersNum.end(), mersenne_engine);

    uint8_t maxTeamPlayers =
        (numPlayers % numTeams == 0) ? numPlayers / numTeams : numPlayers / numTeams + 1;
    std::vector<uint8_t> numPlayersPerTeam(numTeams);
    for (uint8_t i = 0, nP = numPlayers, nT = numTeams; i < numPlayersPerTeam.size(); i++) {
        numPlayersPerTeam[i] = nP / nT;
        nP -= numPlayersPerTeam[i];
        nT--;
    }
    std::vector<uint8_t> playerIDs;
    for (uint8_t i = 0, currentTeam = 0; i < numPlayers; i++) {
        // this->teams[currentTeam].players.emplace_back(players[playersNum[i]].getId());
        playerIDs.emplace_back(players[playersNum[i]].getId());
        players[playersNum[i]].setTeamID(currentTeam);
        if (numPlayersPerTeam[currentTeam] < maxTeamPlayers) {
            players[playersNum[i]].increaseHealth(25.0f);
        }
        if (playerIDs.size() == numPlayersPerTeam[currentTeam]) {
            this->teams.emplace_back(playerIDs, players, ammoCounter);
            playerIDs.clear();
            currentTeam++;
        }
    }
}

void Worms::GameTeams::checkAlive(std::vector<Worms::Player> &players) {
    std::uint8_t teamID{0};
    for (auto &team : this->teams) {
        team.checkAlive(players);
        if (!team.isAlive() && std::find(this->deadTeams.begin(), this->deadTeams.end(), teamID) == this->deadTeams.end()) {
            this->deadTeams.emplace_back(teamID);
        }
        teamID++;
    }
}

```

jun 29, 18 16:28	GameTeams.cpp	Page 2/3
------------------	----------------------	----------

```

bool Worms::GameTeams::endTurn(std::vector<Player> &players) {
    this->checkAlive(players);

    do {
        this->currentTeam = (this->currentTeam + 1) % this->teams.size();
    } while (!this->teams[this->currentTeam].isAlive());

    this->teams[this->currentTeam].endTurn(players);

    if (this->deadTeams.size() >= this->teams.size() - 1) {
        return true;
    } else {
        return false;
    }
}

std::vector<std::uint32_t> Worms::GameTeams::getTotalHealth(std::vector<Worms::Player> &players) {
    uint8_t i{0};
    std::vector<std::uint32_t> teamHealths;
    for (auto &team : this->teams) {
        teamHealths.emplace_back(team.calculateTotalHealth(players));
        i++;
    }
    return std::move(teamHealths);
}

uint8_t Worms::GameTeams::getCurrentPlayerID() {
    return this->teams[this->currentTeam].getCurrentPlayerID();
}

uint8_t Worms::GameTeams::getCurrentTeamID() {
    return this->currentTeam;
}

uint8_t Worms::GameTeams::getWinner() {
    std::uint8_t winner{0};
    for (auto &team : this->teams) {
        if (team.isAlive()) {
            return winner;
        }
        winner++;
    }
    return winner;
}

std::uint8_t Worms::GameTeams::getTeamQuantity() const {
    return (std::uint8_t) this->teams.size();
}

Worms::Team &Worms::GameTeams::getCurrentTeam() {
    return this->teams[this->currentTeam];
}

void Worms::GameTeams::weaponUsed(const Worm::WeaponID weaponID) {
    this->teams[this->currentTeam].weaponUsed(weaponID);
}

void Worms::GameTeams::serialize(IO::GameStateMsg &msg) const {
    this->teams[this->currentTeam].serialize(msg);
}

```

jun 29, 18 16:28

GameTeams.cpp

Page 3/3

```
void Worms::GameTeams::kill(uint8_t team, std::vector<Worms::Player> &players) {  
    this->teams[team].kill(players);  
}
```

jun 29, 18 16:28

GameTeams.h

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#ifndef INC_4_WORMS_TEAMS_H
#define INC_4_WORMS_TEAMS_H

#include <vector>
#include "Player.h"
#include "Team.h"

namespace Worms {
class GameTeams {
public:
    GameTeams() = default;
    ~GameTeams(){};
    void makeTeams(std::vector<Player> &players, uint8_t numTeams,
                  const std::map<Worm::WeaponID, std::int16_t> &ammoCounter);
    void checkAlive(std::vector<Player> &players);
    bool endTurn(std::vector<Player> &players);
    uint8_t getCurrentPlayerID();
    std::uint8_t getTeamQuantity() const;
    uint8_t getCurrentTeamID();
    Team &getCurrentTeam();
    std::uint8_t getWinner();
    std::vector<std::uint32_t> getTotalHealth(std::vector<Worms::Player> &players);
    void weaponUsed(const Worm::WeaponID weaponID);
    void serialize(IO::GameStateMsg &msg) const;

    void kill(uint8_t team, std::vector<Player> &players);

private:
    std::vector<Team> teams;
    std::vector<std::uint8_t> deadTeams;
    uint8_t currentTeam{0};
};
}

#endif // INC_4_WORMS_TEAMS_H

```

jun 29, 18 16:28	GameTurn.cpp	Page 1/2
<pre>// // Created by rodrigo on 10/06/18. // #include "GameTurn.h" #include "Config/Config.h" #include "GameStateMsg.h" #include "GameStates/ImpactOnCourse.h" #include "GameStates/PlayerShot.h" #include "GameStates/StartTurn.h" Worms::GameTurn::GameTurn(Observer &game) : game(game) { this->state = std::shared_ptr<GameTurnState>(new StartTurn()); this->state->addObserver(&this->game); } void Worms::GameTurn::playerShot(Worm::WeaponID weaponID) { this->stateID = GameTurnStateID::PlayerShot; this->newState = true; switch (weaponID) { case Worm::WeaponID::WMortar: this->bulletFragments = ::Game::Config::getInstance().getMortarFragm entQuantity(); break; case Worm::WeaponID::WCluster: this->bulletFragments = ::Game::Config::getInstance().getClusterFrag mentQuantity(); break; case Worm::WeaponID::WAerial: this->bulletFragments = ::Game::Config::getInstance().getAerialAttac kMissileQuantity(); break; default: break; } } void Worms::GameTurn::endTurn() { this->state->endTurn(*this); } void Worms::GameTurn::wormHit(uint8_t wormId) { this->state->wormHit(*this, wormId); } void Worms::GameTurn::explosion() { if (this->stateID != GameTurnStateID::ImpactOnCourse) { this->stateID = GameTurnStateID::ImpactOnCourse; this->state = std::shared_ptr<GameTurnState>(new ImpactOnCourse(this->bu lletFragments)); this->state->addObserver(&this->game); } this->state->explosion(); } void Worms::GameTurn::wormEndHit(uint8_t wormId) { this->state->wormEndHit(*this, wormId); } void Worms::GameTurn::wormDrowning(uint8_t wormId) { this->state->wormDrowning(*this, wormId); }</pre>		

jun 29, 18 16:28	GameTurn.cpp	Page 2/2
<pre> } void Worms::GameTurn::wormDrowned(uint8_t wormId) { this->state->wormDrowned(*this, wormId); } void Worms::GameTurn::restart() { this->stateID = GameTurnStateID::StartTurn; this->newState = true; this->bulletFragments = 1; } void Worms::GameTurn::update(float dt) { if (this->newState) { switch (this->stateID) { case GameTurnStateID::StartTurn: this->state = std::shared_ptr<GameTurnState>(new StartTurn()); break; case GameTurnStateID::PlayerShot: this->state = std::shared_ptr<GameTurnState>(new PlayerShot()); break; case GameTurnStateID::ImpactOnCourse: break; } this->state->addObserver(&this->game); this->newState = false; } this->state->update(dt); } void Worms::GameTurn::wormFalling(uint8_t wormId) { this->state->wormFalling(wormId); } void Worms::GameTurn::wormLanded(uint8_t wormId) { this->state->wormLanded(wormId); } void Worms::GameTurn::wormDead() { this->state->wormDead(); } void Worms::GameTurn::wormDying() { this->state->wormDying(); } void Worms::GameTurn::playerDisconnected(uint8_t wormId) { this->state->wormDisconnectedDying(wormId); } void Worms::GameTurn::playerDisconnectedDead(uint8_t wormId) { this->state->wormDisconnectedDead(wormId); } </pre>		

jun 29, 18 16:28

GameTurn.h

Page 1/1

```

//
// Created by rodrigo on 10/06/18.
//

#ifndef INC_4_WORMS_GAMETURN_H
#define INC_4_WORMS_GAMETURN_H

#include <memory>
#include "GameStates/GameTurnState.h"
#include "Subject.h"

namespace Worms {
enum class GameTurnStateID { StartTurn, PlayerShot, ImpactOnCourse };
class Game;
class GameTurn : public Subject {
public:
    GameTurn(Observer &game);
    ~GameTurn() override = default;

    void playerShot(Worm::WeaponID weaponID);
    void endTurn();
    void wormHit(uint8_t wormId);
    void explosion();
    void wormEndHit(uint8_t wormId);
    void wormDrowning(uint8_t wormId);
    void wormDrowned(uint8_t wormId);
    void restart();
    void update(float dt);
    void wormFalling(uint8_t wormId);
    void wormLanded(uint8_t wormId);
    void wormDead();
    void wormDying();
    void playerDisconnected(uint8_t wormId);
    void playerDisconnectedDead(uint8_t wormId);

private:
    std::shared_ptr<GameTurnState> state{nullptr};
    Observer &game;
    GameTurnStateID stateID;
    bool newState{false};
    uint8_t bulletFragments{1};
};
}

#endif // INC_4_WORMS_GAMETURN_H

```

jun 29, 18 16:28

GameTurnState.cpp

Page 1/1

```
//  
// Created by rodrigo on 10/06/18.  
//  
#include <algorithm>  
#include "GameTurnState.h"  
  
Worms::GameTurnState::GameTurnState() {}  
  
void Worms::GameTurnState::wormFalling(uint8_t wormId) {  
    this->wormsFalling.emplace_back(wormId);  
}  
  
void Worms::GameTurnState::wormLanded(uint8_t wormId) {  
    this->wormsFalling.erase(  
        std::remove(this->wormsFalling.begin(), this->wormsFalling.end(), wormId  
    ),  
        this->wormsFalling.end());  
}  
  
void Worms::GameTurnState::wormDead() {  
    this->wormsDying--;  
}  
  
void Worms::GameTurnState::wormDying() {  
    this->wormsDying++;  
}  
  
uint8_t Worms::GameTurnState::getWormToFollow() const {  
    return this->wormToFollow;  
}
```

jun 29, 18 16:28

GameTurnState.h

Page 1/1

```

//
// Created by rodrigo on 10/06/18.
//

#ifndef INC_4_WORMS_GAMETURNSTATE_H
#define INC_4_WORMS_GAMETURNSTATE_H

#include <stdint>
#include <vector>

#include "../libs/Subject.h"
#include "GameStateMsg.h"

namespace Worms {
class GameTurn;
class GameTurnState : public Subject {
public:
    GameTurnState();
    virtual ~GameTurnState() = default;

    virtual void endTurn(GameTurn &gt) = 0;
    virtual void update(float dt) = 0;
    virtual void wormHit(GameTurn &gt, uint8_t wormId) = 0;
    virtual void wormEndHit(GameTurn &gt, uint8_t wormId) = 0;
    virtual void wormDrowning(GameTurn &gt, uint8_t wormId) = 0;
    virtual void wormDrowned(GameTurn &gt, uint8_t wormId) = 0;
    virtual void explosion() = 0;
    virtual void wormFalling(uint8_t wormId);
    virtual void wormLanded(uint8_t wormId);
    virtual void wormDying();
    virtual void wormDead();
    virtual void wormDisconnectedDying(uint8_t wormId) = 0;
    virtual void wormDisconnectedDead(uint8_t wormId) = 0;
    virtual uint8_t getWormToFollow() const;

protected:
    std::vector<uint8_t> wormsFalling;
    std::vector<uint8_t> wormsDrowning;
    uint8_t wormsDying{0};
    std::vector<uint8_t> wormsDisconnectedDying;
    uint8_t wormToFollow{0};
};
}

#endif // INC_4_WORMS_GAMETURNSTATE_H

```

jun 29, 18 16:28

Girder.cpp

Page 1/1

#include "Girder.h"

```
Worms::Girder::Girder(const Worms::GirderData &data, Worms::Physics &physics)
    : PhysicsEntity(EntityID::EtGirder), angle(data.angle) {
    b2PolygonShape poly;

    b2BodyDef bdef;
    bdef.type = b2_staticBody;
    bdef.position.Set(0.0f, 0.0f);
    b2Body *staticBody = physics.createBody(bdef);

    b2FixtureDef fixture;
    fixture.density = 1;
    fixture.shape = &poly;

    poly.SetAsBox(data.length / 2, data.height / 2, b2Vec2(data.pos.x, data.pos.
y),
                data.angle * (PI / 180.0f));
    staticBody->CreateFixture(&fixture);

    staticBody->SetUserData(this);
}

Worms::Girder::Girder(Worms::Girder &&other) noexcept :
    PhysicsEntity(other.id), angle(other.angle){
    this->handlingContact = other.handlingContact;
    this->numObservers = other.numObservers;
    this->observers = std::move(other.observers);
}
```

jun 29, 18 16:28

Girder.h

Page 1/1

```
#ifndef GIRDER_H_
#define GIRDER_H_

#include "Physics.h"
#include "PhysicsEntity.h"
#include "Stage.h"

namespace Worms {
class Girder : public PhysicsEntity {
public:
    const float angle;

    Girder(const Worms::GirderData &data, Physics &physics);
    Girder(Girder &copy) = delete;
    Girder(Girder &&other) noexcept;
    ~Girder() = default;
};
} // namespace Worms

#endif
```

jun 26, 18 2:39

Grenade.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#include "Grenade.h"
#include "../Player.h"

Weapon::Grenade::Grenade(float angle)
    : Worms::Weapon(Game::Config::getInstance().getGreenGrenadeConfig(), Worms::W
eaponID::WGrenade,
    angle) {
    this->powerChargeTime = Game::Config::getInstance().getPowerChargeTime();
}

void Weapon::Grenade::update(float dt) {
    if (this->increaseShotPower) {
        if (this->shotPower < this->config.maxShotPower) {
            this->shotPower += dt / this->powerChargeTime * this->config.maxShot
Power;
        }
    }
}

void Weapon::Grenade::startShot(Worms::Player *player) {
    this->increaseShotPower = true;
}

void Weapon::Grenade::endShot() {
    this->increaseShotPower = false;
    this->shotPower = 0;
}

void Weapon::Grenade::setTimeout(uint8_t time) {
    this->timeLimit = time;
}

std::list<Worms::Bullet> Weapon::Grenade::onExplode(const Worms::Bullet &bullet,
    Worms::Physics &physics) {
    return std::move(std::list<Worms::Bullet>());
}

void Weapon::Grenade::positionSelected(Worms::Player &p, Math::Point<float> poin
t) {}

```

jun 26, 18 2:39

Grenade.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#ifndef __GRENADE_H__
#define __GRENADE_H__

#include "Weapon.h"

namespace Weapon {
class Grenade : public Worms::Weapon {
public:
    Grenade(float angle);
    ~Grenade() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &bullet,
                                     Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    float powerChargeTime{0.0f};
};
} // namespace Weapon

#endif //__GRENADE_H__

```

jun 26, 18 2:39	Hit.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 28/05/18 */ #include "Hit.h" #include "../Player.h" Worms::Hit::Hit() : State(Worm::StateID::Hit) {} void Worms::Hit::update(Worms::Player &p, float dt, b2Body *body) { /* * when the worm lands (there was a collision between the worm and the * girder) it has to change its state to still, and take an impulse * of equal absolute value and different sign of the impulse taken in * hit stage (remember, the worm has a friction coefficient 0). * * In the y-axis there will be no impulse because its velocity was * cancelled because of the collision with the girder. */ if (p.isOnGround()) { this->timeElapsed += dt; if (this->timeElapsed > 0.7f) { float32 mass = body->GetMass(); b2Vec2 previousVel = body->GetLinearVelocity(); b2Vec2 impulses = {mass * (0.0f - previousVel.x), 0.0f}; body->ApplyLinearImpulseToCenter(impulses, true); p.notify(p, Event::EndHit); p.setState(Worm::StateID::Land); } } else { this->timeElapsed = 0.0f; } } void Worms::Hit::moveRight(Worms::Player &p) {} void Worms::Hit::moveLeft(Worms::Player &p) {} void Worms::Hit::jump(Worms::Player &p) {} void Worms::Hit::stopMove(Worms::Player &p) {} void Worms::Hit::backFlip(Worms::Player &p) {} void Worms::Hit::bazooka(Worms::Player &p) {} void Worms::Hit::pointUp(Worms::Player &p) {} void Worms::Hit::pointDown(Worms::Player &p) {} void Worms::Hit::startShot(Worms::Player &p) {} void Worms::Hit::endShot(Worms::Player &p) {} void Worms::Hit::grenade(Worms::Player &p) {} void Worms::Hit::cluster(Worms::Player &p) {} void Worms::Hit::mortar(Worms::Player &p) {} </pre>		

jun 26, 18 2:39	Hit.cpp	Page 2/2
<pre> void Worms::Hit::banana(Worms::Player &p) {} void Worms::Hit::holy(Worms::Player &p) {} void Worms::Hit::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::Hit::aerialAttack(Worms::Player &p) {} void Worms::Hit::dynamite(Worms::Player &p) {} void Worms::Hit::teleport(Worms::Player &p) {} void Worms::Hit::baseballBat(Worms::Player &p) {} </pre>		

jun 26, 18 2:39

Hit.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 28/05/18
 */

#ifndef __Hit_H__
#define __Hit_H__

#include <cstdint>
#include "PlayerState.h"

namespace Worms {
class Hit : public State {
public:
    Hit();
    ~Hit() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
};
} // namespace Worms

#endif //__Hit_H__

```

jun 26, 18 7:40

Holy.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#include "Holy.h"
#include "../Player.h"

Weapon::Holy::Holy(float angle)
: Worms::Weapon(Game::Config::getInstance().getHolyConfig(), Worm::WeaponID:
:WHoly, angle) {
    this->powerChargeTime = Game::Config::getInstance().getPowerChargeTime();
}

void Weapon::Holy::update(float dt) {
    if (this->increaseShotPower) {
        if (this->shotPower < this->config.maxShotPower) {
            this->shotPower += dt / this->powerChargeTime * this->config.maxShot
Power;
        }
    }
}

void Weapon::Holy::startShot(Worms::Player *player) {
    this->increaseShotPower = true;
}

void Weapon::Holy::endShot() {
    this->increaseShotPower = false;
    this->shotPower = 0;
}

void Weapon::Holy::setTimeout(uint8_t time) {
    this->timeLimit = time;
}

std::list<Worms::Bullet> Weapon::Holy::onExplode(const Worms::Bullet &bullet,
Worms::Physics &physics) {
    return std::move(std::list<Worms::Bullet>());
}

void Weapon::Holy::positionSelected(Worms::Player &p, Math::Point<float> point)
{}

```

jun 26, 18 2:39

Holy.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#ifndef __Holy_H__
#define __Holy_H__

#include "Weapon.h"

namespace Weapon {
class Holy : public Worms::Weapon {
public:
    Holy(float angle);
    ~Holy() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &bullet,
                                     Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    float powerChargeTime{0.0f};
};
} // namespace Weapon

#endif //__Holy_H__

```

jun 29, 18 16:28	ImpactOnCourse.cpp	Page 1/2
<pre>// // Created by rodrigo on 10/06/18. // #include <algorithm> #include <iostream> #include "GameStateMsg.h" #include "ImpactOnCourse.h" void Worms::ImpactOnCourse::endTurn(GameTurn &gt) { if (this->impactEnded && this->wormsFalling.size() == 0 && this->wormsDrowning.size() == 0 && !this->wormsDying && this->wormsDisconnectedDying.size() == 0) { this->notify(*this, Event::TurnEnded); } } void Worms::ImpactOnCourse::wormHit(GameTurn &gt, uint8_t wormId) { this->wormsStillHit.emplace_back(wormId); this->wormsHit.emplace_back(wormId); if (this->wormToFollow != this->wormsStillHit[0]) { this->wormToFollow = this->wormsStillHit[0]; this->notify(*this, Event::NewWormToFollow); } } void Worms::ImpactOnCourse::wormEndHit(Worms::GameTurn &gt, uint8_t wormId) { this->wormsStillHit.erase(std::remove(this->wormsStillHit.begin(), this->wormsStillHit.end(), wormId), this->wormsStillHit.end()); if (this->wormToFollow == wormId) { this->wormToFollow = this->wormsStillHit[0]; this->notify(*this, Event::NewWormToFollow); } } void Worms::ImpactOnCourse::wormDrowning(Worms::GameTurn &gt, uint8_t wormId) { this->wormsDrowning.emplace_back(wormId); this->wormLanded(wormId); if (this->wormsStillHit.size() == 0) { if (this->wormToFollow != this->wormsDrowning[0]) { this->wormToFollow = this->wormsDrowning[0]; this->notify(*this, Event::NewWormToFollow); } } } void Worms::ImpactOnCourse::wormDrowned(Worms::GameTurn &gt, uint8_t wormId) { this->wormsDrowning.erase(std::remove(this->wormsDrowning.begin(), this->wormsDrowning.end(), wormId), this->wormsDrowning.end()); if (this->wormsStillHit.size() == 0) { if (this->wormToFollow != this->wormsDrowning[0]) { this->wormToFollow = this->wormsDrowning[0]; this->notify(*this, Event::NewWormToFollow); } } } </pre>		

jun 29, 18 16:28	ImpactOnCourse.cpp	Page 2/2
<pre>std::vector<uint8_t> &Worms::ImpactOnCourse::getWormsHit() { return this->wormsHit; } void Worms::ImpactOnCourse::impactNotEnded() { this->impactEnded = false; } Worms::ImpactOnCourse::ImpactOnCourse(uint8_t bulletFragments) { this->bulletFragments = bulletFragments; } void Worms::ImpactOnCourse::explosion() { this->fragmentExplosions++; } void Worms::ImpactOnCourse::update(float dt) { if (!this->impactEnded) { if (this->wormsStillHit.size() == 0 && this->wormsDrowning.size() == 0 & & this->fragmentExplosions == this->bulletFragments) { this->impactEnded = true; this->notify(*this, Event::ImpactEnd); } } } void Worms::ImpactOnCourse::wormDisconnectedDying(uint8_t wormId) { this->wormsDisconnectedDying.emplace_back(wormId); if (this->wormToFollow != this->wormsDisconnectedDying[0] && this->wormsFalling.size() == 0 && this->wormsDrowning.size() == 0) { this->wormToFollow = this->wormsDisconnectedDying[0]; this->notify(*this, Event::NewWormToFollow); } } void Worms::ImpactOnCourse::wormDisconnectedDead(uint8_t wormId) { this->wormsDisconnectedDying.erase(std::remove(this->wormsDisconnectedDying.begin(), this->wormsDisconnectedDying.end(), wormId), this->wormsDisconnectedDying.end()); if (this->wormToFollow == wormId) { this->wormToFollow = this->wormsDisconnectedDying[0]; this->notify(*this, Event::NewWormToFollow); } } </pre>		

jun 29, 18 16:28

ImpactOnCourse.h

Page 1/1

```

//
// Created by rodrigo on 10/06/18.
//

#ifndef INC_4_WORMS_IMPACTONCOURSE_H
#define INC_4_WORMS_IMPACTONCOURSE_H

#include <GameStateMsg.h>
#include <vector>
#include "../libs/Observer.h"
#include "GameTurnState.h"

namespace Worms {
class ImpactOnCourse : public GameTurnState {
public:
    ImpactOnCourse(uint8_t bulletFragments);
    ~ImpactOnCourse() = default;

    void endTurn(GameTurn &gt) override;
    void update(float dt) override;
    void wormHit(GameTurn &gt, uint8_t wormId) override;
    void wormEndHit(GameTurn &gt, uint8_t wormId) override;
    void wormDrowning(GameTurn &gt, uint8_t wormId) override;
    void wormDrowned(GameTurn &gt, uint8_t wormId) override;
    void explosion() override;
    void wormDisconnectedDying(uint8_t wormId) override;
    void wormDisconnectedDead(uint8_t wormId) override;
    std::vector<uint8_t> &getWormsHit();
    void impactNotEnded();

private:
    std::vector<uint8_t> wormsStillHit;
    std::vector<uint8_t> wormsHit;
    // uint8_t wormToFollow{0};
    bool impactEnded{false};
    uint8_t bulletFragments{0};
    uint8_t fragmentExplosions{0};
};
}

#endif // INC_4_WORMS_IMPACTONCOURSE_H

```

jun 26, 18 2:39	Jumping.cpp	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 20/05/18 */ #include <Box2D/Dynamics/b2Body.h> #include <iostream> #include <vector> #include "../Player.h" #include "Jumping.h" Worms::Jumping::Jumping(GUI::Position p) : State(Worm::StateID::Jumping), startPos(p) {} void Worms::Jumping::update(Worms::Player &p, float dt, b2Body *body) { /* * when the worm lands (there was a collision between the worm and the * girder) it has to changes its state to endJump, and take an impulse * of equal absolute value and different sign of the impulse taken in * startJump stage (remember, the worm has a friction coefficient 0). * * In the y-axis there will be no impulse because its velocity was * cancelled because of the collision with the girder. */ if (p.isOnGround()) { this->timeElapsed += dt; } else { this->timeElapsed = 0.0f; } if (p.isOnGround() this->timeElapsed > 0.2f) { float32 mass = body->GetMass(); b2Vec2 previousVel = body->GetLinearVelocity(); b2Vec2 impulses = {mass * (0.0f - previousVel.x), 0.0f}; body->ApplyLinearImpulseToCenter(impulses, true); p.landDamage(this->startPosition.y - p.getPosition().y); p.setState(Worm::StateID::Land); // p.setState(Worm::StateID::EndJump); } } void Worms::Jumping::moveRight(Worms::Player &p) {} void Worms::Jumping::moveLeft(Worms::Player &p) {} void Worms::Jumping::jump(Worms::Player &p) {} void Worms::Jumping::stopMove(Worms::Player &p) {} void Worms::Jumping::backFlip(Worms::Player &p) {} void Worms::Jumping::bazooka(Worms::Player &p) {} void Worms::Jumping::pointUp(Worms::Player &p) {} void Worms::Jumping::pointDown(Worms::Player &p) {} void Worms::Jumping::startShot(Worms::Player &p) {} void Worms::Jumping::endShot(Worms::Player &p) {} </pre>		

jun 26, 18 2:39	Jumping.cpp	Page 2/2
<pre> void Worms::Jumping::grenade(Worms::Player &p) {} void Worms::Jumping::cluster(Worms::Player &p) {} void Worms::Jumping::mortar(Worms::Player &p) {} void Worms::Jumping::banana(Worms::Player &p) {} void Worms::Jumping::holy(Worms::Player &p) {} void Worms::Jumping::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::Jumping::aerialAttack(Worms::Player &p) {} void Worms::Jumping::dynamite(Worms::Player &p) {} void Worms::Jumping::teleport(Worms::Player &p) {} void Worms::Jumping::baseballBat(Worms::Player &p) {} </pre>		

jun 26, 18 2:39

Jumping.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/05/18
 */

#ifndef __PLAYER_JUMPING_H__
#define __PLAYER_JUMPING_H__

#include <Box2D/Dynamics/b2Body.h>
#include <Camera.h>

#include "PlayerState.h"

namespace Worms {
class Jumping : public State {
public:
    Jumping(GUI::Position p);
    ~Jumping() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    virtual void pointUp(Player &p) override;
    virtual void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    GUI::Position startPosition;
};
} // namespace Worms

#endif //__PLAYER_JUMPING_H__

```

jun 29, 18 16:28	Land.cpp	Page 1/2
<pre>// // Created by rodrigo on 3/06/18. // #include "Land.h" #include "../Config/Config.h" #include "../Player.h" #include "PlayerState.h" Worms::Land::Land() : State(Worm::StateID::Land), landTime(Game::Config::getInstance().getLandTime()) {} void Worms::Land::update(Worms::Player &p, float dt, b2Body *body) { this->timeElapsed += dt; if (this->timeElapsed > this->landTime) { p.notify(p, Event::WormLanded); if (p.health <= 0) { p.notify(p, Event::Dying); p.setState(Worm::StateID::Die); } else { p.setState(Worm::StateID::Still); } } } void Worms::Land::moveRight(Worms::Player &p) {} void Worms::Land::moveLeft(Worms::Player &p) {} void Worms::Land::jump(Worms::Player &p) {} void Worms::Land::stopMove(Worms::Player &p) {} void Worms::Land::backFlip(Worms::Player &p) {} void Worms::Land::bazooka(Worms::Player &p) {} void Worms::Land::pointUp(Worms::Player &p) {} void Worms::Land::pointDown(Worms::Player &p) {} void Worms::Land::startShot(Worms::Player &p) {} void Worms::Land::endShot(Worms::Player &p) {} void Worms::Land::grenade(Worms::Player &p) {} void Worms::Land::cluster(Worms::Player &p) {} void Worms::Land::mortar(Worms::Player &p) {} void Worms::Land::banana(Worms::Player &p) {} void Worms::Land::holy(Worms::Player &p) {} void Worms::Land::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::Land::aerialAttack(Worms::Player &p) {} void Worms::Land::dynamite(Worms::Player &p) {}</pre>		

jun 29, 18 16:28	Land.cpp	Page 2/2
<pre>void Worms::Land::teleport(Worms::Player &p) {} void Worms::Land::baseballBat(Worms::Player &p) {}</pre>		

jun 26, 18 2:39

Land.h

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#ifndef INC_4_WORMS_LAND_H
#define INC_4_WORMS_LAND_H

#include <stdint>
#include "PlayerState.h"

namespace Worms {
class Land : public State {
public:
    Land();
    ~Land() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    float landTime;
};
} // namespace Worms

#endif // INC_4_WORMS_LAND_H

```

jun 29, 18 16:28	Lobbies.cpp	Page 1/2
<pre>// // Created by rodrigo on 15/06/18. // #include <iostream> #include <yaml-cpp/yaml.h> #include "GamesGetter.h" #include "Lobbies.h" void Worms::Lobbies::createGame(int playerID, std::vector<Observer *> lobbyObservers, uint8_t levelSelected) { std::lock_guard<std::mutex> lock(this->mutex); this->lobbies.emplace_back(playerID, this->idLobby, lobbyObservers, this->levels[levelSelected], this->levelsInfo[levelSelected]); this->idLobby++; } void Worms::Lobbies::getGames(GamesGetter &getter) { std::lock_guard<std::mutex> lock(this->mutex); getter(this->lobbies); } void Worms::Lobbies::joinGame(int gameID, int playerID, Observer *lobbyObserver) { std::lock_guard<std::mutex> lock(this->mutex); auto it = this->lobbies.begin(); while ((*it).getID() != gameID && it != this->lobbies.end()) { it++; } (*it).addLobbyObserver(lobbyObserver); (*it).joinGame(playerID); } std::list<Worms::Lobby> &Worms::Lobbies::getLobbies() { return this->lobbies; } Worms::Lobbies::Lobbies(const std::vector<IO::LevelData> &levels) : levels(levels) {} const std::vector<IO::LevelInfo> &Worms::Lobbies::getLevels() { return this->levelsInfo; } const IO::LevelData & Worms::Lobbies::getLevelData(uint8_t levelSelected) { return this->levels[levelSelected]; } void Worms::Lobbies::configure() { uint8_t id{0}; for (auto &level : this->levels) { YAML::Node data = YAML::LoadFile(level.levelPath); std::string name = data["name"].as<std::string>(); uint8_t playersQuantity = static_cast<uint8_t>(data["numPlayers"].as<int>()); IO::LevelInfo levelInfo{id, name, playersQuantity}; this->levelsInfo.emplace_back(levelInfo); id++; } } Worms::Lobbies::~Lobbies() {}</pre>		

jun 29, 18 16:28	Lobbies.cpp	Page 2/2

jun 29, 18 16:28

Lobbies.h

Page 1/1

```

//
// Created by rodrigo on 15/06/18.
//

#ifndef INC_4_WORMS_LOBBIES_H
#define INC_4_WORMS_LOBBIES_H

#include <list>
#include <mutex>

#include "GamesGetter.h"
#include "Lobby.h"
#include "Observer.h"

namespace Worms {
    class Lobbies {
    public:
        explicit Lobbies(const std::vector<IO::LevelData> &levels);
        ~Lobbies();
        void configure();
        void createGame(int playerId, std::vector<Observer *> lobbyObservers, ui
nt8_t levelSelected);
        void getGames(GamesGetter &getter);
        void joinGame(int gameId, int playerId, Observer *lobbyObserver);
        const std::vector<IO::LevelInfo> &getLevels();
        const IO::LevelData & getLevelData(uint8_t levelSelected);
        std::list<Lobby> &getLobbies();

    private:
        std::mutex mutex;
        std::list<Lobby> lobbies;
        uint8_t idLobby{0};
        const std::vector<IO::LevelData> &levels;
        std::vector<IO::LevelInfo> levelsInfo;
    };
}

#endif //INC_4_WORMS_LOBBIES_H

```

jun 29, 18 16:28

Lobby.cpp

Page 1/2

```
//
// Created by rodrigo on 16/06/18.
//

#include <iostream>
#include <string>

#include "Lobby.h"
#include "Stage.h"
#include "Game.h"

/** Copio por Â¿posible race condition?
 */
Worms::Lobby::Lobby(int playerID, std::uint8_t id, std::vector<Observer *> obs,
const IO::LevelData level,
const IO::LevelInfo levelInfo) :
    id(id),
    level(level),
    levelInfo(levelInfo) {
    for (auto *lobbyObserver : obs) {
        this->obs.emplace_back(lobbyObserver);
        this->addObserver(lobbyObserver);
    }
    this->joinGame(playerID);
}

void Worms::Lobby::joinGame(int playerID) {
    // std::lock_guard<std::mutex> lock(this->mutex);
    this->playerIDs.emplace_back(playerID);
    this->actualPlayers++;
    this->notify(*this, Event::NewPlayer);
    if (this->actualPlayers == levelInfo.playersQuantity) {
        this->notify(*this, Event::StartGame);
        std::uint8_t i{0};
        for (auto *obs : this->obs) {
            if (i != 0) {
                this->removeObserver(obs);
            }
            i++;
        }
        this->gameStarted = true;
    }
}

const IO::LevelInfo & Worms::Lobby::getLevelInfo() const{
    return this->levelInfo;
}

std::uint8_t Worms::Lobby::getActualPlayers() const{
    return this->actualPlayers;
}

const std::vector<int> &Worms::Lobby::getPlayerIDs() const{
    return this->playerIDs;
}

std::uint8_t Worms::Lobby::getID() const{
    return this->id;
}

void Worms::Lobby::addPlayerSocket(CommunicationSocket &&player) {
```

jun 29, 18 16:28

Lobby.cpp

Page 2/2

```
    this->players.emplace_back(std::move(player));
}

Worms::Lobby::Lobby(Worms::Lobby &&other) noexcept :
    id(other.id),
    level(other.level),
    levelInfo(other.levelInfo) {
    if (this != &other){
        this->actualPlayers = other.actualPlayers;
        this->playerIDs = std::move(other.playerIDs);
        this->players = std::move(other.players);
    }
}

void Worms::Lobby::run() {
    try {
        while (!this->finished) {
            if (this->gameStarted) {
                for (std::uint8_t i = 0; i < levelInfo.playersQuantity; i++) {
                    char buffer[1];
                    buffer[0] = i;
                    this->players[i].send(buffer, sizeof(buffer));
                }
                Worms::Game game{Worms::Stage::fromFile(this->level.levelPath),
this->players};
                game.start();
                this->notify(*this, Event::EndGame);
                this->gameStarted = false;
                this->finished = true;
            }
        }
    } catch (std::exception &e){
        if (!this->finished){
            std::cerr << "In Lobby::run()" << std::endl;
            std::cerr << e.what() << std::endl;
        }
    } catch (...) {
        std::cerr << "Unkown error in Lobby::run()" << std::endl;
    }
}

void Worms::Lobby::stop() {
    this->finished = true;
    for(auto &player: this->players){
        player.shutdown();
    }
}

bool Worms::Lobby::itsOver() {
    return this->finished;
}

void Worms::Lobby::addLobbyObserver(Observer *lobbyObserver) {
    this->obs.emplace_back(lobbyObserver);
    this->addObserver(lobbyObserver);
}

bool Worms::Lobby::started() {
    return this->gameStarted;
}
```

jun 29, 18 16:28

Lobby.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_LOBBY_H
#define INC_4_WORMS_LOBBY_H

#include <stdint-gcc.h>
#include <string>
#include <vector>
#include <mutex>
#include <GameStateMsg.h>

#include "CommunicationSocket.h"
#include "Subject.h"
#include "Thread.h"

namespace Worms {
    class Lobby : public Thread, public Subject {
    public:
        Lobby(int playerID, std::uint8_t id, std::vector<Observer *> obs, const
IO::LevelData level,
            const IO::LevelInfo levelInfo);
        Lobby(Lobby &&other) noexcept;
        Lobby(Lobby &copy) = delete;

        void run() override;
        void stop() override;
        bool itsOver();

        void joinGame(int playerID);
        const IO::LevelInfo & getLevelInfo() const;
        std::uint8_t getActualPlayers() const;
        const std::vector<int> &getPlayerIDs() const;
        std::uint8_t getID() const;
        void addLobbyObserver(Observer *lobbyObserver);
        bool started();
        void addPlayerSocket(CommunicationSocket &&player);

    private:
        std::mutex mutex;
        const std::uint8_t id;
        std::uint8_t actualPlayers{0};
        std::vector<int> playerIDs;
        std::vector<CommunicationSocket> players;
        std::vector<Observer *> obs;
        const IO::LevelData level;
        const IO::LevelInfo levelInfo;

        bool finished{false};
        bool gameStarted{false};
    };
}

#endif //INC_4_WORMS_LOBBY_H

```

jun 29, 18 16:28

LobbyJoiner.cpp

Page 1/2

```
//
// Created by rodrigo on 19/06/18.
//

#include <GameStateMsg.h>
#include <iostream>
#include "LobbyJoiner.h"

Worms::LobbyJoiner::LobbyJoiner(Worms::Lobbies &lobbies, IO::Stream<IO::ServerInternalMsg> &serverInput) :
    lobbies(lobbies.getLobbies()),
    serverInput(serverInput) {}

void Worms::LobbyJoiner::run() {
    try{
        while (!this->finished) {
            IO::ServerInternalMsg msg;
            if (this->serverInput.pop(msg)) {
                this->handleServerInput(msg);
            }
        }
    } catch (std::exception &e){
        if(!this->finished){
            std::cerr << "In LobbyJoiner::run()" << std::endl;
            std::cerr << e.what() << std::endl;
        }
    } catch (...){
        std::cerr << "Unknown error in LobbyJoiner::run()" << std::endl;
    }

    this->killLobbies();
}

void Worms::LobbyJoiner::stop() {
    this->finished = true;
}

void Worms::LobbyJoiner::handleServerInput(IO::ServerInternalMsg &msg) {
    switch (msg.action) {
        case IO::ServerInternalAction::lobbyFinished: {
            std::list<Lobby>::iterator lobbyIt;
            lobbyIt = this->lobbies.begin();
            while (lobbyIt != this->lobbies.end()) {
                if (lobbyIt->itsOver()) {
                    lobbyIt->join();
                    lobbyIt = this->lobbies.erase(lobbyIt);
                } else {
                    lobbyIt++;
                }
            }
            break;
        }
        case IO::ServerInternalAction::quit: {
            this->finished = true;
            break;
        }
    }
}

void Worms::LobbyJoiner::killLobbies(){
    for (auto &lobby: this->lobbies){
```

jun 29, 18 16:28

LobbyJoiner.cpp

Page 2/2

```
        if (lobby.started()) {
            lobby.stop();
            lobby.join();
        }
    }
    this->lobbies.erase(this->lobbies.begin(), this->lobbies.end());
}
```

jun 29, 18 16:28

LobbyJoiner.h

Page 1/1

```

//
// Created by rodrigo on 19/06/18.
//

#ifndef INC_4_WORMS_LOBBYJOINER_H
#define INC_4_WORMS_LOBBYJOINER_H

#include <GameStateMsg.h>
#include <Stream.h>
#include "Thread.h"
#include "Lobbies.h"

namespace Worms {
    class LobbyJoiner : public Thread {
    public:
        explicit LobbyJoiner(Worms::Lobbies &lobbies, IO::Stream<IO::ServerInternalMsg> &serverInput);

        void run() override;
        void stop() override;

    private:
        std::list<Lobby> &lobbies;
        IO::Stream<IO::ServerInternalMsg> &serverInput;
        bool finished{false};

        void handleServerInput(IO::ServerInternalMsg &msg);

        void killLobbies();
    };
}

#endif //INC_4_WORMS_LOBBYJOINER_H

```

jun 29, 18 16:28

main.cpp

Page 1/2

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#include <signal.h>
#include <unistd.h>
#include <cstdlib>
#include <iostream>
#include <string>
#include <thread>
#include <vector>

#include "CommunicationSocket.h"
#include "Game.h"
#include "ServerSocket.h"
#include "GameLobby.h"

static volatile bool quit = false;

/**
 * @brief Signal handler.
 *
 * @param _ unused.
 */
static void _signal_handler(int _) {
    quit = true;
}

/**
 * @brief Thread handler that signals the Game to exit.
 *
 * @param game
 */
static void _exit_handler(Worms::Game &game) {
    while (!quit) {
        usleep(100000);
    }
    game.exit();
}

int main(int argc, const char *argv[]) {
    if (argc != 2) {
        std::cout << "Usage: ./server PORT" << std::endl;
        return EXIT_FAILURE;
    }

    try {
        /* sets a signal handler to exit the program gracefully */
        signal(SIGINT, _signal_handler);
        signal(SIGTERM, _signal_handler);

        std::string port(argv[1]);
        Worms::GameLobby gameLobby{port};

        gameLobby.start();
        char quit{0};
        while (quit != 'q') {
            std::cin >> quit;
        }

        gameLobby.stop();
    }
}

```

jun 29, 18 16:28

main.cpp

Page 2/2

```

        gameLobby.join();

    } catch (std::exception &e) {
        std::cerr << "In main()" << std::endl;
        std::cerr << e.what() << std::endl;
        return 1;
    } catch (...) {
        std::cerr << "Unkown error in main thread" << std::endl;
        return 1;
    }

    return EXIT_SUCCESS;
}

```


jun 26, 18 2:39	Mortar.cpp	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 03/06/18 */ #include "Mortar.h" #include "../Player.h" Weapon::Mortar::Mortar(float angle) : Worms::Weapon(Game::Config::getInstance().getMortarConfig(), Worm::WeaponID::WMortar, angle), fragmentConfig(Game::Config::getInstance().getMortarFragmentConfig()) {} void Weapon::Mortar::update(float dt) { if (this->increaseShotPower) { if (this->shotPower >= this->config.maxShotPower) { this->shotPower = this->config.maxShotPower; } else { this->shotPower++; } } } void Weapon::Mortar::startShot(Worms::Player *player) { this->increaseShotPower = true; } void Weapon::Mortar::endShot() { this->increaseShotPower = false; this->shotPower = 0; } void Weapon::Mortar::setTimeout(uint8_t time) {} std::list<Worms::Bullet> Weapon::Mortar::onExplode(const Worms::Bullet &mainBullet, Worms::Physics &physics) { uint8_t fragmentQuantity = Game::Config::getInstance().getMortarFragmentQuantity(); Math::Point<float> p = mainBullet.getPosition(); Worms::BulletInfo bulletInfo = {this->fragmentConfig.dmgInfo, p, this->fragmentConfig.minAngle, (float)this->fragmentConfig.maxShotPower, this->fragmentConfig.bulletRadius * 6, this->fragmentConfig.restitution, this->fragmentConfig.friction, this->fragmentConfig.explosionInitialTimeout, Event::Explode, this->fragmentConfig.bulletRadius, this->fragmentConfig.bulletDampingRatio, this->config.windAffected}; std::list<Worms::Bullet> ret; for (int i = 0; i < fragmentQuantity; i++) { bulletInfo.angle = i * this->fragmentConfig.angleStep + this->fragmentConfig.minAngle; ret.emplace_back(bulletInfo, physics, Worm::WeaponID::WFragment); } return std::move(ret); </pre>		

jun 26, 18 2:39	Mortar.cpp	Page 2/2
<pre> } void Weapon::Mortar::positionSelected(Worms::Player &p, Math::Point<float> point) {} </pre>		

jun 26, 18 2:39

Mortar.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 03/06/18
 */

#ifndef __Mortar_H__
#define __Mortar_H__

#include "Weapon.h"

namespace Weapon {
class Mortar : public Worms::Weapon {
public:
    Mortar(float angle);
    ~Mortar() override = default;
    void update(float dt) override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &bullet,
                                      Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;

private:
    const Config::Weapon &fragmentConfig;
};
} // namespace Weapon

#endif // __Mortar_H__

```

jun 26, 18 2:39

P2PWeapon.cpp

Page 1/1

```
/*  
 * Created by Federico Manuel Gomez Peter.  
 * date: 22/06/18  
 */
```

```
#include "P2PWeapon.h"
```

jun 26, 18 2:39

P2PWeapon.h

Page 1/1

```
/*  
 * Created by Federico Manuel Gomez Peter.  
 * date: 22/06/18  
 */
```

```
#ifndef __P2PWeapon_H__  
#define __P2PWeapon_H__
```

```
#include <Direction.h>  
#include <Point.h>
```

```
#include "BulletConfig.h"
```

```
namespace Config {  
struct P2PWeapon {  
    Bullet::DamageInfo dmgInfo;  
    Worm::Direction direction;  
    Math::Point<float> position;  
    float angle;  
};  
} // namespace Config
```

```
#endif //__P2PWeapon_H__
```

jun 26, 18 2:39

Physics.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#include "Physics.h"

Worms::Physics::Physics(b2Vec2 gravity, float timeStep)
: timeStep(timeStep),
  gravity(gravity),
  world(this->gravity),
  contactEventListener(new ContactEventListener) {
    this->world.SetContactListener(this->contactEventListener.get());
}

/**
 * @brief Updates the physics engine.
 *
 * @param dt Seconds elapsed since last call to this function.
 */
void Worms::Physics::update(float dt) {
    this->accumTime += dt;

    /* updates the physics engine */
    for (int i = 0; i < 5 && this->accumTime > this->timeStep; i++) {
        this->world.Step(this->timeStep, this->vIterations, this->pIterations);
        this->accumTime -= this->timeStep;
    }
}

/**
 * @brief Creates a new physical body.
 *
 * @param bodyDef Body definition.
 * @return new body.
 */
b2Body* Worms::Physics::createBody(b2BodyDef& bodyDef) {
    return this->world.CreateBody(&bodyDef);
}

```

jun 29, 18 16:28

PhysicsEntity.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 26/05/18
 */

#include "PhysicsEntity.h"

Worms::PhysicsEntity::PhysicsEntity(Worms::EntityID id) : id(id) {}

Worms::EntityID Worms::PhysicsEntity::getEntityId() {
    return this->id;
}

Worms::PhysicsEntity::PhysicsEntity(Worms::PhysicsEntity &&other){
    this->id = other.id;
    this->handlingContact = other.handlingContact;

    other.handlingContact = false;
}
```

jun 29, 18 16:28

PhysicsEntity.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 26/05/18
 */

#ifndef PHYSICS_ENTITY_H_
#define PHYSICS_ENTITY_H_

#include "Box2D/Box2D.h"

#include "Subject.h"

namespace Worms {
enum EntityID { EtWorm, EtBullet, Sensor, EtGirder };
class PhysicsEntity : public Subject {
public:
    explicit PhysicsEntity(EntityID id);
    PhysicsEntity(PhysicsEntity &&other);
    PhysicsEntity(PhysicsEntity &copy) = delete;

    virtual EntityID getEntityId();
    virtual void startContact(Worms::PhysicsEntity *physicsEntity) {}
    virtual void startContact(Worms::PhysicsEntity *physicsEntity, b2Contact &contact) {}
    virtual void endContact(Worms::PhysicsEntity *physicsEntity) {}
    virtual void endContact(Worms::PhysicsEntity *physicsEntity, b2Contact &contact) {}
    virtual void contactWith(PhysicsEntity &physicsEntity, b2Contact &contact) {}

protected:
    EntityID id;
    bool handlingContact{false};
};
} // namespace Worms

#endif

```

jun 26, 18 2:39

Physics.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 18/05/18
 */

#ifndef __Physics_H__
#define __Physics_H__

#include "Box2D/Box2D.h"

#include <memory.h>
#include "ContactEventListener.h"

namespace Worms {
class Physics {
public:
    Physics(b2Vec2 gravity, float timeStep);
    ~Physics() = default;
    void update(float dt);
    b2Body *createBody(b2BodyDef &bodyDef);

private:
    float timeStep;
    float accumTime{0.0f};
    b2Vec2 gravity;
    b2World world;
    std::shared_ptr<ContactEventListener> contactEventListener;
    int32 vIterations{6};
    int32 pIterations{2};
};
} // namespace Worms

#endif // __Physics_H__
```


jun 29, 18 16:28	Player.cpp	Page 1/11
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #include <Box2D/Box2D.h> #include <iostream> #include "Direction.h" #include "Girder.h" #include "Physics.h" #include "Player.h" #include "Weapons/AerialAttack.h" #include "Weapons/Banana.h" #include "Weapons/BaseballBat.h" #include "Weapons/Bazooka.h" #include "Weapons/Cluster.h" #include "Weapons/Dynamite.h" #include "Weapons/Grenade.h" #include "Weapons/Holy.h" #include "Weapons/Mortar.h" #include "Weapons/Teleport.h" #include "Weapons/Weapon.h" #include "WormStates/BackFlipping.h" #include "WormStates/Batting.h" #include "WormStates/Dead.h" #include "WormStates/Die.h" #include "WormStates/Drowning.h" #include "WormStates/EndBackFlip.h" #include "WormStates/EndJump.h" #include "WormStates/Falling.h" #include "WormStates/Hit.h" #include "WormStates/Jumping.h" #include "WormStates/Land.h" #include "WormStates/Sliding.h" #include "WormStates/StartBackFlip.h" #include "WormStates/StartJump.h" #include "WormStates/Still.h" #include "WormStates/Teleported.h" #include "WormStates/Teleporting.h" #include "WormStates/Walk.h" #include "Weapons/WeaponNone.h" #define CONFIG Game::Config::getInstance() Worms::Player::Player(Physics &physics) : PhysicsEntity(Worms::EntityID::EtWorm), physics(physics), waterLevel(CONFI G.getWaterLevel()) { /* creates 2 bodies so players cannot move each other */ this->body = this->createBody(b2_dynamicBody); this->body_kinematic = this->createBody(b2_kinematicBody); /* creates the sensor as a circle */ b2CircleShape sensorShape; sensorShape.m_radius = PLAYER_HEIGHT / 4; sensorShape.m_p.Set(0.0f, -PLAYER_HEIGHT / 4 - 0.2); /* allocated in heap because it's address shouldn't change */ this->footSensor = new TouchSensor{*this->body, sensorShape}; this->footSensor->ignore(*this); this->setState(Worm::StateID::Falling); </pre>		

jun 29, 18 16:28	Player.cpp	Page 2/11
<pre> this->weapon = std::shared_ptr<Worms::Weapon>(new Worms::Weapon::Bazooka(0.0f)); } Worms::Player::~Player() { delete this->footSensor; } /** * @brief "Not equal" operator. * * @param other Other instance to compare. * @return true if not equal. */ bool Worms::Player::operator!=(const Player &other) { return !(*this == other); } /** * @brief Comparisson operator. * * @param other Other instance to compare. * @return true if equal. */ bool Worms::Player::operator==(const Player &other) { return (this->id == other.id) && (this->teamID == other.teamID); } /** * @brief Handles player-entity contact. * * @param other Other player that made contact. * @param contact box2D collision contact. */ void Worms::Player::contactWith(PhysicsEntity &entity, b2Contact &contact) { if (entity.getEntityId() == Worms::EntityID::EtGirder) { Worms::Girder &girder = dynamic_cast<Worms::Girder &>(entity); if (std::abs(girder.angle) > PI / 4.0f) { this->lastGroundNormal = contact.GetManifold()->localNormal; } else { this->lastGroundNormal = {0.0f, 0.1f}; } } if (entity.getEntityId() != Worms::EntityID::EtWorm) { return; } /* checks if it's the player itself */ if (&entity == this) { /* checks if it's the kinematic and dynamic bodies colliding */ if (contact.GetFixtureA()->GetBody()->GetType() != contact.GetFixtureB()->GetBody()->GetType()) { contact.SetEnabled(false); } } } void Worms::Player::update(float dt) { /* sets the kinematic body to the position of the dynamic body */ this->body_kinematic->SetTransform(this->body->GetTransform().p, this->body- >GetAngle()); } </pre>		

jun 29, 18 16:28

Player.cpp

Page 3/11

```

    this->state->update(*this, dt, this->body);
    this->weapon->update(dt);

    if (this->getPosition().y <= this->waterLevel && this->getStateId() != Worm::StateID::Dead &&
        this->getStateId() != Worm::StateID::Drowning) {
        this->health = 0;
        if (this->getStateId() == Worm::StateID::Hit) {
            this->notify(*this, Event::EndHit);
        }
        this->setState(Worm::StateID::Drowning);
        this->notify(*this, Event::Drowning);
    } else if (this->isOnGround()) {
        /* checks if the ground slope is too tilted */
        try {
            b2Vec2 normal = this->getGroundNormal();
            float slope = std::abs(std::atan2(normal.y, normal.x));
            if ((slope < PI / 4.0f) || (slope > (PI * 3.0f) / 4.0f)) {
                if (this->getStateId() == Worm::StateID::Hit) {
                    this->notify(*this, Event::EndHit);
                }
                this->setState(Worm::StateID::Sliding);
                return;
            }
        } catch (const Exception &e) {
        }
    }

    /**
     * @brief Whether the player is touching the ground or not.
     *
     * @return true is touching the ground.
     */
    bool Worms::Player::isOnGround() const {
        return this->footSensor->isActive();
    }

    void Worms::Player::setPosition(const Math::Point<float> &new_pos) {
        this->body->SetTransform(b2Vec2(new_pos.x, new_pos.y), body->GetAngle());
    }

    /**
     * @brief Returns a unit vector with the direction normal to the floor where the
     * player is standing.
     *
     * @return b2Vec2 Floor normal.
     */
    b2Vec2 Worms::Player::getGroundNormal() const {
        for (auto &contact : *this->footSensor) {
            if (contact.first->getEntityId() == Worms::EntityID::EtGirder) {
                return this->lastGroundNormal;
            }
        }
        throw Exception("No ground normal");
    }

    void Worms::Player::startContact(Worms::PhysicsEntity *physicsEntity, b2Contact &contact) {}

    Math::Point<float> Worms::Player::getPosition() const {
        const b2Vec2 &pos = this->body->GetPosition();

```

jun 29, 18 16:28

Player.cpp

Page 4/11

```

        return Math::Point<float>{pos.x, pos.y};
    }

    Worm::StateID Worms::Player::getStateId() const {
        return this->state->getState();
    }

    void Worms::Player::handleState(IO::PlayerMsg pi) {
        switch (pi.input) {
            case IO::PlayerInput::moveLeft:
                this->state->moveLeft(*this);
                break;
            case IO::PlayerInput::moveRight:
                this->state->moveRight(*this);
                break;
            case IO::PlayerInput::startJump:
                this->state->jump(*this);
                break;
            case IO::PlayerInput::startBackFlip:
                this->state->backFlip(*this);
                break;
            case IO::PlayerInput::stopMove:
                this->state->stopMove(*this);
                break;
            case IO::PlayerInput::bazooka:
                this->state->bazooka(*this);
                break;
            case IO::PlayerInput::grenade:
                this->state->grenade(*this);
                break;
            case IO::PlayerInput::cluster:
                this->state->cluster(*this);
                break;
            case IO::PlayerInput::mortar:
                this->state->mortar(*this);
                break;
            case IO::PlayerInput::banana:
                this->state->banana(*this);
                break;
            case IO::PlayerInput::holy:
                this->state->holy(*this);
                break;
            case IO::PlayerInput::moveNone:
                break;
            case IO::PlayerInput::pointUp:
                this->state->pointUp(*this);
                break;
            case IO::PlayerInput::pointDown:
                this->state->pointDown(*this);
                break;
            case IO::PlayerInput::startShot:
                this->state->startShot(*this);
                break;
            case IO::PlayerInput::endShot:
                this->state->endShot(*this);
                break;
            case IO::PlayerInput::timeout1:
                this->state->setTimeout(*this, 1);
                break;
            case IO::PlayerInput::timeout2:
                this->state->setTimeout(*this, 2);
                break;

```

jun 29, 18 16:28

Player.cpp

Page 5/11

```

        case IO::PlayerInput::timeout3:
            this->state->setTimeout(*this, 3);
            break;
        case IO::PlayerInput::timeout4:
            this->state->setTimeout(*this, 4);
            break;
        case IO::PlayerInput::timeout5:
            this->state->setTimeout(*this, 5);
            break;
        case IO::PlayerInput::positionSelected:
            this->weapon->positionSelected(*this, pi.position);
            break;
        case IO::PlayerInput::aerialAttack:
            this->state->aerialAttack(*this);
            break;
        case IO::PlayerInput::dynamite:
            this->state->dynamite(*this);
            break;
        case IO::PlayerInput::baseballBat:
            this->state->baseballBat(*this);
            break;
        case IO::PlayerInput::teleport:
            this->state->teleport(*this);
            break;
        default:
            break;
    }
}

void Worms::Player::setState(Worm::StateID stateID) {
    if (this->state == nullptr || this->state->getState() != stateID) {
        /* creates the right state type */
        this->body->SetType(b2_dynamicBody);
        switch (stateID) {
            case Worm::StateID::Still:
                // this->body->SetType(b2_staticBody);
                this->state = std::shared_ptr<State>(new Still());
                break;
            case Worm::StateID::Walk:
                this->state = std::shared_ptr<State>(new Walk());
                break;
            case Worm::StateID::StartJump:
                this->state = std::shared_ptr<State>(new StartJump());
                break;
            case Worm::StateID::Jumping:
                this->state = std::shared_ptr<State>(new Jumping(this->getPosition()));
                break;
            case Worm::StateID::EndJump:
                this->state = std::shared_ptr<State>(new EndJump());
                break;
            case Worm::StateID::StartBackFlip:
                this->state = std::shared_ptr<State>(new StartBackFlip());
                break;
            case Worm::StateID::BackFlipping:
                this->state = std::shared_ptr<State>(new BackFlipping(this->getPosition()));
                break;
            case Worm::StateID::EndBackFlip:
                this->state = std::shared_ptr<State>(new EndBackFlip());
                break;
            case Worm::StateID::Falling:

```

jun 29, 18 16:28

Player.cpp

Page 6/11

```

        this->state = std::shared_ptr<State>(new Falling(this->getPosition()));
        break;
        case Worm::StateID::Land:
            this->state = std::shared_ptr<State>(new Land());
            break;
        case Worm::StateID::Batting:
            this->state = std::shared_ptr<State>(new Batting());
            break;
        case Worm::StateID::Teleporting:
            this->state = std::shared_ptr<State>(new Teleporting(this->teleportPosition()));
            break;
        case Worm::StateID::Teleported:
            this->state = std::shared_ptr<State>(new Teleported());
            break;
        case Worm::StateID::Hit:
            this->state = std::shared_ptr<State>(new Hit());
            break;
        case Worm::StateID::Die:
            this->state = std::shared_ptr<State>(new Die());
            break;
        case Worm::StateID::Drowning:
            this->state = std::shared_ptr<State>(new Drowning());
            break;
        case Worm::StateID::Dead:
            this->state = std::shared_ptr<State>(new Dead());
            this->body->SetType(b2_staticBody);
            break;
        case Worm::StateID::Sliding:
            this->notify(*this, Event::WormFalling);
            this->state = std::shared_ptr<State>(new Sliding());
            break;
    }
}

std::list<Worms::Bullet> Worms::Player::getBullets() {
    return std::move(this->bullets);
}

void Worms::Player::acknowledgeDamage(Config::Bullet::DamageInfo damageInfo,
    Math::Point<float> epicenter) {
    if (this->getStateId() != Worm::StateID::Dead) {
        double distanceToEpicenter = this->getPosition().distance(epicenter);
        if (distanceToEpicenter <= damageInfo.radius) {
            this->body->SetType(b2_dynamicBody);
            double inflictedDamage =
                (1.0f - (distanceToEpicenter / (damageInfo.radius * 1.01f))) * damageInfo.damage;
            this->health -= inflictedDamage;

            Math::Point<float> positionToEpicenter = this->getPosition() - epicenter;
            float xImpactDirection = (positionToEpicenter.x > 0) - (positionToEpicenter.x < 0);
            float yImpactDirection = (positionToEpicenter.y > 0) - (positionToEpicenter.y < 0);
            float32 mass = this->body->GetMass();
            b2Vec2 impulses = {
                mass * float32(inflictedDamage) * xImpactDirection * damageInfo.
                impulseDampingRatio,

```

jun 29, 18 16:28	Player.cpp	Page 7/11
<pre> mass * float32(inflictedDamage) * yImpactDirection * damageInfo.impulseDampingRatio); b2Vec2 position = this->body->GetWorldCenter(); this->body->ApplyLinearImpulse(impulses, position, true); this->notify(*this, Event::Hit); this->setState(Worm::StateID::Hit); this->health = (this->health < 0) ? 0 : this->health; } } void Worms::Player::acknowledgeDamage(const Config::P2PWeapon &info, Math::Point<float> shooterPosition, Worm::Direction shooterDirection) { if (this->getStateId() != Worm::StateID::Dead) { if ((shooterDirection == Worm::Direction::right && this->getPosition().x - shooterPosition.x > 0) (shooterDirection == Worm::Direction::left && this->getPosition().x - shooterPosition.x < 0)) { double distanceToTheWeapon = this->getPosition().distance(info.posit ion); if (distanceToTheWeapon <= info.dmgInfo.radius && distanceToTheWeapo n > 0) { this->body->SetType(b2_dynamicBody); this->health -= info.dmgInfo.damage; this->health = (this->health < 0) ? 0 : this->health; float32 mass = this->body->GetMass(); Math::Point<float> direction{0, 0}; direction.x = info.dmgInfo.radius * cos(info.angle * PI / 180.0f); direction.y = info.dmgInfo.radius * sin(info.angle * PI / 180.0f); Math::Point<float> positionToShooter = this->getPosition() - sho oterPosition; float xImpactDirection = (positionToShooter.x > 0) - (positionTo Shooter.x < 0); float yImpactDirection = (direction.y > 0) - (direction.y < 0); b2Vec2 impulses = {mass * float32(info.dmgInfo.damage) * directi on.x * xImpactDirection * info.dmgInfo.impulseDa mpingRatio, mass * float32(info.dmgInfo.damage) * directi on.y * yImpactDirection * info.dmgInfo.impulseDa mpingRatio}; this->body->ApplyLinearImpulse(impulses, this->body->GetWorldCen ter(), true); this->notify(*this, Event::Hit); this->setState(Worm::StateID::Hit); } } } float Worms::Player::getWeaponAngle() const { return this->weapon->getAngle(); } const Worm::WeaponID &Worms::Player::getWeaponID() const { return this->weapon->getWeaponID(); } </pre>		

jun 29, 18 16:28	Player.cpp	Page 8/11
<pre> } void Worms::Player::setWeapon(const Worm::WeaponID &id) { // keep the last angle float lastAngle = this->weapon->getAngle(); this->weapon = this->team->getWeapon(id); this->weapon->setAngle(lastAngle); this->isP2PWeapon = this->weapon->isP2PWeapon(); } void Worms::Player::increaseWeaponAngle() { this->weapon->increaseAngle(); } void Worms::Player::decreaseWeaponAngle() { this->weapon->decreaseAngle(); } void Worms::Player::startShot() { this->weapon->startShot(this); } void Worms::Player::endShot() { if (this->weapon->getWeaponID() != Worm::WeaponID::WTeleport && this->weapon->getWeaponID() != Worm::WeaponID::WAerial && this->weapon->getWeaponID() != Worm::WeaponID::WNone) { if (!this->isP2PWeapon) { Math::Point<float> position = this->getPosition(); float safeNonContactDistance = sqrt((PLAYER_WIDTH / 2) * (PLAYER_WID TH / 2) + (PLAYER_HEIGHT / 2) * (PLAYER_HE IGHT / 2)) + 0.1; BulletInfo info = this->weapon->getBulletInfo(); info.point = position; info.safeNonContactDistance = safeNonContactDistance; if (this->direction == Worm::Direction::right) { if (info.angle < 0.0f) { info.angle += 360.0f; } } else { info.angle = 180.0f - info.angle; } this->bullets.emplace_back(info, this->physics, this->weapon->getWea ponID()); this->weapon->endShot(); this->notify(*this, Event::Shot); } else { this->setState(Worm::StateID::Batting); this->notify(*this, Event::P2PWeaponUsed); } this->team->weaponUsed(this->getWeaponID()); } } void Worms::Player::endShot(std::list<Worms::Bullet> &bullets) { this->bullets = std::move(bullets); this->notify(*this, Event::Shot); this->team->weaponUsed(this->getWeaponID()); } void Worms::Player::setTeamID(uint8_t team) { this->teamID = team; } </pre>		

jun 29, 18 16:28	Player.cpp	Page 9/11
<pre> } void Worms::Player::increaseHealth(float extraPoints) { // this->health += (percentage / 100.0f) * this->health; // 25% more this->health += extraPoints; // 25 points more } uint8_t Worms::Player::getTeam() const { return this->teamID; } void Worms::Player::setId(uint8_t id) { this->id = id; } uint8_t Worms::Player::getId() const { return this->id; } void Worms::Player::setWeaponTimeout(uint8_t time) { this->weapon->setTimeout(time); } void Worms::Player::landDamage(float yDistance) { if (yDistance > CONFIG.getSafeFallDistance()) { this->health -= (yDistance > CONFIG.getMaxFallDamage()) ? CONFIG.getMaxFallDamage() : yDistance; this->health = (this->health < 0) ? 0 : this->health; if (this->health > 0) { this->notify(*this, Event::DamageOnLanding); } } } /** * @brief Creates a player's body with the given type. * * @param type Body type. * @return Created body. */ b2Body *Worms::Player::createBody(b2BodyType type) { /* the players consists of a rectangle as the upper part of the body and a circle for the * bottom */ b2BodyDef bodyDef; bodyDef.type = type; bodyDef.position.Set(0.0f, 0.0f); bodyDef.fixedRotation = true; b2Body *new_body = this->physics.createBody(bodyDef); b2PolygonShape shape; shape.SetAsBox(PLAYER_WIDTH / 2, PLAYER_HEIGHT / 4, b2Vec2{0.0f, PLAYER_HEIGHT / 4}, 0.0f); /* creates the upper square */ b2FixtureDef fixture; fixture.shape = &shape; fixture.density = 1.0f; fixture.restitution = 0.1f; fixture.friction = 1.0f; new_body->CreateFixture(&fixture); </pre>		

jun 29, 18 16:28	Player.cpp	Page 10/11
<pre> /* creates the bottom circle */ b2CircleShape bottom; bottom.m_radius = PLAYER_HEIGHT / 4; bottom.m_p.Set(0.0f, -PLAYER_HEIGHT / 4); fixture.shape = &bottom; new_body->CreateFixture(&fixture); new_body->SetUserData(this); return new_body; } std::list<Worms::Bullet> Worms::Player::onExplode(const Bullet &b, Physics &physics) { return std::move(this->weapon->onExplode(b, physics)); } void Worms::Player::reset() { this->weapon->endShot(); /* * If the weapon has no more ammunition, returns weaponNone */ this->setWeapon(this->getWeaponID()); this->bullets.erase(this->bullets.begin(), this->bullets.end()); } Worms::Physics &Worms::Player::getPhysics() { return this->physics; } const std::shared_ptr<Worms::Weapon> Worms::Player::getWeapon() const { return this->weapon; } void Worms::Player::setTeam(Worms::Team *team) { this->team = team; } Worms::Player::Player(Worms::Player &&player) noexcept: PhysicsEntity(std::move(player)), physics(player.physics), waterLevel(player.waterLevel) { this->body = player.body; this->body_kinematic = player.body_kinematic; this->footSensor = player.footSensor; this->state = player.state; this->weapon = player.weapon; this->team = player.team; this->id = player.id; this->bullets = std::move(player.bullets); player.body = nullptr; player.body_kinematic = nullptr; player.footSensor = nullptr; player.state = nullptr; player.weapon = nullptr; player.team = 0; player.id = 0; } void Worms::Player::die() { this->setState(Worm::StateID::Die); } </pre>		

jun 29, 18 16:28

Player.cpp

Page 11/11

```
this->health = 0;  
this->dyingDisconnected = true;  
this->notify(*this, Event::DyingDueToDisconnection);  
}
```

jun 29, 18 16:28	Player.h	Page 1/3
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 18/05/18 */ #ifndef __PLAYER_H__ #define __PLAYER_H__ #define PLAYER_WIDTH 0.8f #define PLAYER_HEIGHT 2.0f #include <list> #include "Config/Config.h" #include "Config/P2PWeapon.h" #include "Direction.h" #include "GameStateMsg.h" #include "Physics.h" #include "Point.h" #include "Stream.h" #include "Team.h" #include "TouchSensor.h" #include "Weapons/Bullet.h" #include "Weapons/Weapon.h" #include "WormStates/PlayerState.h" enum class PlayerState { movingRight, movingLeft, still }; namespace Worms { class Player : public PhysicsEntity { public: Worm::Direction direction{Worm::Direction::left}; Worm::Direction lastWalkDirection; std::uint16_t health{0}; Math::Point<float> teleportPosition{0.0f, 0.0f}; bool dyingDisconnected{false}; explicit Player(Physics &physics); Player(Player &&player) noexcept; Player(Player &copy) = delete; ~Player(); /* contact handlers */ virtual void contactWith(PhysicsEntity &other, b2Contact &contact); bool isOnGround() const; /** * Updates its state, its weapon * @param dt */ void update(float dt); void serialize(IO::Stream<IO::GameStateMsg> &s) const {} /** * @brief moves the player to newPos position * @param newPos */ void setPosition(const Math::Point<float> &newPos); b2Vec2 getGroundNormal() const; </pre>		

jun 29, 18 16:28	Player.h	Page 2/3
<pre> void startContact(Worms::PhysicsEntity *physicsEntity, b2Contact &contact); /** * @brief asks box2D from current position. * @return */ Math::Point<float> getPosition() const; /** * @brief given playerInput, changes its state (or its weapon) accordingly * @param pi */ void handleState(IO::PlayerMsg pi); const std::shared_ptr<Worms::Weapon> getWeapon() const; Worm::StateID getStateID() const; void setState(Worm::StateID stateID); float getWeaponAngle() const; const Worm::WeaponID &getWeaponID() const; void setWeapon(const Worm::WeaponID &id); /** * @brief delegates on its weapon the action of increase the angle, if * the weapon handles it. */ void increaseWeaponAngle(); /** * @brief delegates on its weapon the action of decrease the angle, if * the weapon handles it. */ void decreaseWeaponAngle(); /** * @brief delegates on its weapon the action of starting a shot, increasing * its powerShot if it handles it */ void startShot(); /** * @brief creates a bullet that needs to be moved using getBullet() */ void endShot(); void acknowledgeDamage(Config::Bullet::DamageInfo damageInfo, Math::Point<float> epicenter); void acknowledgeDamage(const Config::P2PWeapon &info, Math::Point<float> shooterPosition, Worm::Direction shooterDirection); void landDamage(float yDistance); void setTeamID(uint8_t team); void setTeam(Worms::Team *team); void increaseHealth(float extraPoints); uint8_t getTeam() const; void setId(uint8_t id); uint8_t getId() const; Physics &getPhysics(); void setWeaponTimeout(uint8_t time); /** * Moves the bullets to the caller (the Game) * @return bullets */ std::list<Bullet> getBullets(); /** * Resets the weapon's powershot and erase every possible bullet * inside his container. */ void reset(); </pre>		

jun 29, 18 16:28

Player.h

Page 3/3

```

    * calls weapon's onExplode and get new bullets if it is necessary.
    */
    std::list<Bullet> onExplode(const Bullet &bullet, Physics &physics);

    bool operator!=(const Player &other);
    bool operator==(const Player &other);

    void endShot(std::list<Worms::Bullet> &bullets);
    void die();

private:
    b2Body *createBody(b2BodyType type);

    b2Body *body{nullptr};
    b2Body *body_kinematic{nullptr};
    TouchSensor *footSensor;

    std::shared_ptr<Worms::State> state{nullptr};
    std::shared_ptr<Worms::Weapon> weapon{nullptr};
    Physics &physics;
    const int waterLevel;
    uint8_t teamID;
    uint8_t id;
    std::list<Bullet> bullets;
    bool isP2PWeapon{false};
    b2Vec2 lastGroundNormal{0.0f, 0.0f};
    Team *team{nullptr};
};
} // namespace Worms

#endif // __PLAYER_H__

```


jun 29, 18 16:28

PlayerShot.cpp

Page 1/1

```
//  
// Created by rodrigo on 10/06/18.  
//  
  
#include "PlayerShot.h"  
  
void Worms::PlayerShot::endTurn(GameTurn &gt) {}  
  
void Worms::PlayerShot::wormHit(GameTurn &gt, uint8_t wormId) {}  
  
void Worms::PlayerShot::wormEndHit(Worms::GameTurn &gt, uint8_t wormId) {}  
  
void Worms::PlayerShot::wormDrowning(Worms::GameTurn &gt, uint8_t wormId) {}  
  
void Worms::PlayerShot::wormDrowned(Worms::GameTurn &gt, uint8_t wormId) {}  
  
Worms::PlayerShot::PlayerShot() {}  
  
void Worms::PlayerShot::explosion() {}  
  
void Worms::PlayerShot::update(float dt) {}  
  
void Worms::PlayerShot::wormDisconnectedDying(uint8_t wormId) {}  
  
void Worms::PlayerShot::wormDisconnectedDead(uint8_t wormId) {}
```

jun 29, 18 16:28

PlayerShot.h

Page 1/1

```

//
// Created by rodrigo on 10/06/18.
//

#ifndef INC_4_WORMS_PLAYERSHOT_H
#define INC_4_WORMS_PLAYERSHOT_H

#include "../libs/Observer.h"
#include "GameTurnState.h"

namespace Worms {
class PlayerShot : public GameTurnState {
public:
    PlayerShot();
    ~PlayerShot() = default;

    void endTurn(GameTurn &gt) override;
    void update(float dt) override;
    void wormHit(GameTurn &gt, uint8_t wormId) override;
    void wormEndHit(GameTurn &gt, uint8_t wormId) override;
    void wormDrowning(GameTurn &gt, uint8_t wormId) override;
    void wormDrowned(GameTurn &gt, uint8_t wormId) override;
    void explosion() override;
    void wormDisconnectedDying(uint8_t wormId) override;
    void wormDisconnectedDead(uint8_t wormId) override;
};
}

#endif // INC_4_WORMS_PLAYERSHOT_H

```

jun 26, 18 2:39

PlayerState.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 20/05/18
 */

#include "PlayerState.h"
#include "GameStateMsg.h"

Worms::State::State(Worm::StateID id) : stateID(id) {}

Worm::StateID Worms::State::getState() const {
    return this->stateID;
}
```

jun 26, 18 2:39

PlayerState.h

Page 1/1

```

#ifndef _PLAYERSTATE_H
#define _PLAYERSTATE_H

#include <Box2D/Common/b2Math.h>
#include <Box2D/Dynamics/b2Body.h>
#include <vector>

#include "GameStateMsg.h"

namespace Worms {
class Player;
class State {
public:
    explicit State(Worm::StateID id);
    virtual ~State() = default;
    virtual void update(Player &p, float dt, b2Body *body) = 0;
    virtual void moveRight(Player &p) = 0;
    virtual void moveLeft(Player &p) = 0;
    virtual void jump(Player &p) = 0;
    virtual void setTimeout(Player &p, uint8_t time) = 0;

    virtual void bazooka(Player &p) = 0;
    virtual void grenade(Player &p) = 0;
    virtual void cluster(Player &p) = 0;
    virtual void mortar(Player &p) = 0;
    virtual void banana(Player &p) = 0;
    virtual void holy(Player &p) = 0;
    virtual void aerialAttack(Player &p) = 0;
    virtual void dynamite(Player &p) = 0;
    virtual void baseballBat(Player &p) = 0;
    virtual void teleport(Player &p) = 0;

    virtual void startShot(Player &p) = 0;
    virtual void endShot(Player &p) = 0;
    virtual void backFlip(Player &p) = 0;
    virtual void stopMove(Player &p) = 0;
    virtual void pointUp(Player &p) = 0;
    virtual void pointDown(Player &p) = 0;
    virtual Worm::StateID getState() const;

protected:
    Worm::StateID stateID;
    std::vector<float> impulses{0.0f, 0.0f};
};
}

#endif // _PLAYERSTATE_H

```

jun 26, 18 2:39

ServerSocket.cpp

Page 1/2

```

/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#include <netdb.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#include <cstring>

#include "ErrorMessages.h"
#include "Exception.h"
#include "ServerSocket.h"

ServerSocket::ServerSocket(const char *port) {
    this->bindAndListen(port);
}

void ServerSocket::bindAndListen(const char *port) {
    int status = 0;
    int option_value = 1;
    bool is_bound = false;
    /*
     * inicializo el bloque de memoria de addrinfo,
     * lo configuro para que result sea una lista de
     * address pertenecientes a IPv4, y que sean TCP.
     */
    struct addrinfo hints = {AI_PASSIVE, AF_INET, SOCK_STREAM, 0, 0, nullptr, nullptr, nullptr};
    struct addrinfo *result, *ptr;

    status = getaddrinfo(nullptr, port, &hints, &result);
    if (status != 0) {
        throw Exception(ERR_MSG_SOCKET_INVALID_PORT, port, gai_strerror(status));
    }
    /*
     * Recorro los resultados posibles, hasta poder bindear
     */
    for (ptr = result; ptr != nullptr && !is_bound; ptr = ptr->ai_next) {
        this->fd = ::socket(ptr->ai_family, ptr->ai_socktype, ptr->ai_protocol);
        /*
         * si la creaci3n del socket falla, no debo hacer nada mas
         * en el ciclo (ya que no se abrio ningun fd)
         */
        if (this->fd == -1) {
            continue;
        }
        /*
         * Del ejemplo del echoserver, se obtuvo la forma de
         * configurar la reutilizaci3n de la direcci3n
         * que no se encuentre disponible por un TIME_WAIT.
         * si la configuraci3n falla, debo liberar
         * el socket (segun la documentaci3n y el ejemplo
         * que se encuentra en el manual de getaddrinfo)
         */
        status =
            setsockopt(this->fd, SOL_SOCKET, SO_REUSEADDR, &option_value, sizeof
(option_value));
        if (status == -1) {

```

jun 26, 18 2:39

ServerSocket.cpp

Page 2/2

```

        this->close();
        continue;
    }
    /*
     * Si logro bindear, salgo del ciclo, sino, cierro el socket
     * y pruebo en el siguiente resultado.
     */
    status = bind(this->fd, result->ai_addr, result->ai_addrlen);
    if (status == -1) {
        this->close();
    } else {
        is_bound = true;
    }
    freeaddrinfo(result);

    if (!is_bound) {
        throw Exception(ERR_MSG_SOCKET_BINDING, port);
    }

    status = listen(this->fd, 20);
    if (status == -1) {
        throw Exception(ERR_MSG_SOCKET_LISTEN, strerror(errno));
    }
}

CommunicationSocket ServerSocket::accept() {
    int fd = ::accept(this->fd, nullptr, nullptr);
    if (fd == -1) {
        throw Exception(ERR_MSG_SOCKET_ACCEPT, strerror(errno));
    }
    return std::move(CommunicationSocket(fd));
}

```

jun 26, 18 2:39

ServerSocket.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter
 * Date: 02/05/2018.
 */

#ifndef __SERVERSOCKET_H__
#define __SERVERSOCKET_H__

#include <string>

#include "CommunicationSocket.h"
#include "Socket.h"

class ServerSocket : public Socket {
public:
    explicit ServerSocket(const char *port);
    /**
     * Acepta una conexi3n y devuelve un CommunicationSocket por movimiento.
     * @return Socket para comunicacion
     */
    CommunicationSocket accept();
    void bindAndListen(const char *port);
};

#endif //__SERVERSOCKET_H__
```

jun 26, 18 2:39	Sliding.cpp	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 20/05/18 */ #include <iostream> #include <vector> #include "../Player.h" #include "Sliding.h" Worms::Sliding::Sliding() : State(Worm::StateID::Sliding) {} void Worms::Sliding::update(Worms::Player &p, float dt, b2Body *body) { if (!p.isOnGround()) { p.setState(Worm::StateID::Falling); return; } float final_vel{0.0f}; try { b2Vec2 normal = p.getGroundNormal(); float slope = std::abs(std::atan2(normal.y, normal.x)); if ((slope < PI / 4.0f) (slope > (PI * 3.0f) / 4.0f)) { final_vel = 3.0f * normal.x; float impulse = body->GetMass() * (final_vel - body->GetLinearVelocity().x); body->ApplyLinearImpulse(b2Vec2(impulse, 0.0f), body->GetWorldCenter(), true); } else { p.setState(Worm::StateID::Land); } } catch (const Exception &e) {} } void Worms::Sliding::moveRight(Worms::Player &p) {} void Worms::Sliding::moveLeft(Worms::Player &p) {} void Worms::Sliding::jump(Worms::Player &p) {} void Worms::Sliding::stopMove(Worms::Player &p) {} void Worms::Sliding::backFlip(Worms::Player &p) {} void Worms::Sliding::bazooka(Worms::Player &p) {} void Worms::Sliding::pointUp(Worms::Player &p) {} void Worms::Sliding::pointDown(Worms::Player &p) {} void Worms::Sliding::startShot(Worms::Player &p) {} void Worms::Sliding::endShot(Worms::Player &p) {} void Worms::Sliding::grenade(Worms::Player &p) {} void Worms::Sliding::cluster(Worms::Player &p) {} </pre>		

jun 26, 18 2:39	Sliding.cpp	Page 2/2
<pre> void Worms::Sliding::mortar(Worms::Player &p) {} void Worms::Sliding::banana(Worms::Player &p) {} void Worms::Sliding::holy(Worms::Player &p) {} void Worms::Sliding::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::Sliding::aerialAttack(Worms::Player &p) {} void Worms::Sliding::dynamite(Worms::Player &p) {} void Worms::Sliding::teleport(Worms::Player &p) {} void Worms::Sliding::baseballBat(Worms::Player &p) {} </pre>		

jun 26, 18 2:39

Sliding.h

Page 1/1

```

#ifndef _PLAYER_SLIDING_H
#define _PLAYER_SLIDING_H

#include "../Config/Config.h"
#include "PlayerState.h"

namespace Worms {
class Sliding : public State {
public:
    Sliding();
    ~Sliding() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    virtual void pointUp(Player &p) override;
    virtual void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
};
} // namespace Worms

#endif // _PLAYER_SLIDING_H

```


jun 26, 18 2:39	StartBackFlip.cpp	Page 1/2
<pre> /* * Created by Rodrigo. * date: 20/05/18 */ #include "StartBackFlip.h" #include "Direction.h" Worms::StartBackFlip::StartBackFlip() : State(Worm::StateID::StartBackFlip), backflipVelocity(Game::Config::getInstance().getBackflipVelocity()), startJumpTime(Game::Config::getInstance().getStartJumpTime()) {} void Worms::StartBackFlip::update(Worms::Player &p, float dt, b2Body *body) { this->timeElapsed += dt; if (this->timeElapsed >= this->startJumpTime) { if (!this->impulseApplied) { float32 mass = body->GetMass(); b2Vec2 impulses = {mass * this->backflipVelocity.x, mass * this->backflipVelocity.y}; if (p.direction == Worm::Direction::left) { impulses.x *= -1; } /* When the worm jumps, it needs an initial impulse in the y axis * that will never will be applied again. In the x axis, the worms * moves in RUM, so it needs an initial impulse (because his friction * coefficient is 0) and then needs an end impulse, of equal absolute * value and different sign. */ body->ApplyLinearImpulse(impulses, body->GetWorldCenter(), true); this->impulseApplied = true; } else if (!p.isOnGround()) { p.setState(Worm::StateID::BackFlipping); } else if (this->timeElapsed > 0.9f) { p.setState(Worm::StateID::Still); } } } void Worms::StartBackFlip::moveRight(Worms::Player &p) {} void Worms::StartBackFlip::moveLeft(Worms::Player &p) {} void Worms::StartBackFlip::jump(Worms::Player &p) {} void Worms::StartBackFlip::backFlip(Worms::Player &p) {} void Worms::StartBackFlip::stopMove(Worms::Player &p) {} void Worms::StartBackFlip::bazooka(Worms::Player &p) {} void Worms::StartBackFlip::pointUp(Worms::Player &p) {} void Worms::StartBackFlip::pointDown(Worms::Player &p) {} void Worms::StartBackFlip::startShot(Worms::Player &p) {} void Worms::StartBackFlip::endShot(Worms::Player &p) {} void Worms::StartBackFlip::grenade(Worms::Player &p) {} </pre>		

jun 26, 18 2:39	StartBackFlip.cpp	Page 2/2
<pre> void Worms::StartBackFlip::cluster(Worms::Player &p) {} void Worms::StartBackFlip::mortar(Worms::Player &p) {} void Worms::StartBackFlip::banana(Worms::Player &p) {} void Worms::StartBackFlip::holy(Worms::Player &p) {} void Worms::StartBackFlip::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::StartBackFlip::aerialAttack(Worms::Player &p) {} void Worms::StartBackFlip::dynamite(Worms::Player &p) {} void Worms::StartBackFlip::teleport(Worms::Player &p) {} void Worms::StartBackFlip::baseballBat(Worms::Player &p) {} </pre>		

jun 26, 18 2:39

StartBackFlip.h

Page 1/1

```

/*
 * Created by Rodrigo.
 * date: 20/05/18
 */

#ifndef __PLAYER_START_BACK_FLIP_H__
#define __PLAYER_START_BACK_FLIP_H__

#include <stdint-gcc.h>
#include <stdint>
#include "../Config/Config.h"
#include "../Player.h"

namespace Worms {
class StartBackFlip : public State {
public:
    StartBackFlip();
    ~StartBackFlip() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    bool impulseApplied{false};
    const Math::Vector backflipVelocity;
    const float startJumpTime;
};
}

#endif //__PLAYER_START_BACK_FLIP_H__

```

jun 26, 18 2:39	StartJump.cpp	Page 1/2
<pre>// // Created by Gorco on 19/05/18. // #include <iostream> #include "../Config/Config.h" #include "Direction.h" #include "StartJump.h" Worms::StartJump::StartJump() : State(Worm::StateID::StartJump), jumpTime(Game::Config::getInstance().getStartJumpTime()), jumpVelocity(Game::Config::getInstance().getJumpVelocity()) {} void Worms::StartJump::update(Player &p, float dt, b2Body *body) { this->timeElapsed += dt; if (this->timeElapsed >= this->jumpTime) { if (!this->impulseApplied) { float32 mass = body->GetMass(); b2Vec2 impulses = {mass * this->jumpVelocity.x, mass * this->jumpVelocity.y}; if (p.direction == Worm::Direction::left) { impulses.x *= -1; } /* When the worm jumps, it needs an initial impulse in the y axis * that will never will be applied again. In the x axis, the worms * moves in RUM, so it needs an initial impulse (because his friction * coefficient is 0) and then needs an end impulse, of equal absolute * value and different sign. */ body->ApplyLinearImpulse(impulses, body->GetWorldCenter(), true); this->impulseApplied = true; } else if (!p.isOnGround()) { p.setState(Worm::StateID::Jumping); } else if (this->timeElapsed > 0.9f) { p.setState(Worm::StateID::Still); } } } void Worms::StartJump::moveRight(Worms::Player &p) {} void Worms::StartJump::moveLeft(Worms::Player &p) {} void Worms::StartJump::jump(Worms::Player &p) {} void Worms::StartJump::stopMove(Worms::Player &p) {} void Worms::StartJump::backFlip(Worms::Player &p) {} void Worms::StartJump::bazooka(Worms::Player &p) {} void Worms::StartJump::pointUp(Worms::Player &p) {} void Worms::StartJump::pointDown(Worms::Player &p) {} void Worms::StartJump::startShot(Worms::Player &p) {} void Worms::StartJump::endShot(Worms::Player &p) {}</pre>		

jun 26, 18 2:39	StartJump.cpp	Page 2/2
<pre>void Worms::StartJump::grenade(Worms::Player &p) {} void Worms::StartJump::cluster(Worms::Player &p) {} void Worms::StartJump::mortar(Worms::Player &p) {} void Worms::StartJump::banana(Worms::Player &p) {} void Worms::StartJump::holy(Worms::Player &p) {} void Worms::StartJump::setTimeout(Worms::Player &p, uint8_t time) {} void Worms::StartJump::aerialAttack(Worms::Player &p) {} void Worms::StartJump::dynamite(Worms::Player &p) {} void Worms::StartJump::teleport(Worms::Player &p) {} void Worms::StartJump::baseballBat(Worms::Player &p) {}</pre>		

jun 26, 18 2:39

StartJump.h

Page 1/1

```

//
// Created by Gorco on 19/05/18.
//

#ifndef __WORMS_PLAYER_JUMP_RIGHT_H__
#define __WORMS_PLAYER_JUMP_RIGHT_H__

#include <stdint-gcc.h>
#include <cstdint>
#include "../Config/Config.h"
#include "../Player.h"

namespace Worms {
class StartJump : public State {
public:
    StartJump();
    ~StartJump() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    bool impulseApplied{false};
    const float jumpTime;
    const Math::Vector jumpVelocity;
};
} // namespace Worms

#endif // __WORMS_PLAYER_JUMP_RIGHT_H__

```

jun 29, 18 16:28

StartTurn.cpp

Page 1/1

```

//
// Created by rodrigo on 10/06/18.
//

#include <algorithm>

#include "StartTurn.h"

void Worms::StartTurn::endTurn(GameTurn &gt) {
    if (this->wormsFalling.size() == 0 && this->wormsDrowning.size() == 0 && !this->wormsDying && this->wormsDisconnectedDying.size() == 0) {
        this->notify(*this, Event::TurnEnded);
    }
}

void Worms::StartTurn::wormHit(GameTurn &gt, uint8_t wormId) {}

void Worms::StartTurn::wormEndHit(Worms::GameTurn &gt, uint8_t wormId) {}

void Worms::StartTurn::wormDrowning(Worms::GameTurn &gt, uint8_t wormId) {
    this->wormsDrowning.emplace_back(wormId);
    this->wormLanded(wormId);
}

void Worms::StartTurn::wormDrowned(Worms::GameTurn &gt, uint8_t wormId) {
    this->wormsDrowning.erase(
        std::remove(this->wormsDrowning.begin(), this->wormsDrowning.end(), wormId),
        this->wormsDrowning.end());
}

Worms::StartTurn::StartTurn() {}

void Worms::StartTurn::explosion() {}

void Worms::StartTurn::update(float dt) {}

void Worms::StartTurn::wormDisconnectedDying(uint8_t wormId) {
    this->wormsDisconnectedDying.emplace_back(wormId);
    if (this->wormToFollow != this->wormsDisconnectedDying[0] && this->wormsFalling.size() == 0 && this->wormsDrowning.size() == 0) {
        this->wormToFollow = this->wormsDisconnectedDying[0];
        this->notify(*this, Event::NewWormToFollow);
    }
}

void Worms::StartTurn::wormDisconnectedDead(uint8_t wormId) {
    this->wormsDisconnectedDying.erase(
        std::remove(this->wormsDisconnectedDying.begin(), this->wormsDisconnectedDying.end(), wormId),
        this->wormsDisconnectedDying.end());
    if (this->wormToFollow == wormId) {
        this->wormToFollow = this->wormsDisconnectedDying[0];
        this->notify(*this, Event::NewWormToFollow);
    }
}

```

jun 29, 18 16:28

StartTurn.h

Page 1/1

```
//  
// Created by rodrigo on 10/06/18.  
//  
  
#ifndef INC_4_WORMS_STARTTURN_H  
#define INC_4_WORMS_STARTTURN_H  
  
#include "../libs/Observer.h"  
#include "GameTurnState.h"  
  
namespace Worms {  
class StartTurn : public GameTurnState {  
public:  
    StartTurn();  
    ~StartTurn() = default;  
  
    void endTurn(GameTurn &gt) override;  
    void update(float dt) override;  
    void wormHit(GameTurn &gt, uint8_t wormId) override;  
    void wormEndHit(GameTurn &gt, uint8_t wormId) override;  
    void wormDrowning(GameTurn &gt, uint8_t wormId) override;  
    void wormDrowned(GameTurn &gt, uint8_t wormId) override;  
    void explosion() override;  
    void wormDisconnectedDying(uint8_t wormId) override;  
    void wormDisconnectedDead(uint8_t wormId) override;  
};  
}  
  
#endif // INC_4_WORMS_STARTTURN_H
```

jun 26, 18 2:39	Still.cpp	Page 1/2
<pre>// // Created by Gorco on 19/05/18. // #include <stdint> #include <iostream> #include <memory> #include "../Player.h" #include "Still.h" #include "Walk.h" Worms::Still::Still() : State(Worm::StateID::Still) {} void Worms::Still::update(Player &p, float dt, b2Body *body) { float32 mass = body->GetMass(); b2Vec2 vel = body->GetLinearVelocity(); this->impulses[0] = -vel.x * mass; body->ApplyLinearImpulse(b2Vec2(impulses[0], impulses[1]), body->GetWorldCenter(), true); } void Worms::Still::moveRight(Worms::Player &p) { p.direction = Worm::Direction::right; p.setState(Worm::StateID::Walk); } void Worms::Still::moveLeft(Worms::Player &p) { p.direction = Worm::Direction::left; p.setState(Worm::StateID::Walk); } void Worms::Still::stopMove(Worms::Player &p) {} void Worms::Still::jump(Worms::Player &p) { p.notify(p, Event::WormFalling); p.setState(Worm::StateID::StartJump); } void Worms::Still::backFlip(Worms::Player &p) { p.notify(p, Event::WormFalling); p.setState(Worm::StateID::StartBackFlip); } void Worms::Still::bazooka(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WBazooka); } void Worms::Still::pointUp(Worms::Player &p) { p.increaseWeaponAngle(); } void Worms::Still::pointDown(Worms::Player &p) { p.decreaseWeaponAngle(); } void Worms::Still::startShot(Worms::Player &p) { p.startShot(); } void Worms::Still::endShot(Worms::Player &p) {</pre>		

jun 26, 18 2:39	Still.cpp	Page 2/2
<pre> p.endShot(); } void Worms::Still::grenade(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WGrenade); } void Worms::Still::cluster(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WCluster); } void Worms::Still::mortar(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WMortar); } void Worms::Still::banana(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WBanana); } void Worms::Still::holy(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WHoly); } void Worms::Still::setTimeout(Worms::Player &p, uint8_t time) { p.setWeaponTimeout(time); } void Worms::Still::aerialAttack(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WAerial); } void Worms::Still::dynamite(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WDynamite); } void Worms::Still::teleport(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WTeleport); } void Worms::Still::baseballBat(Worms::Player &p) { p.setWeapon(Worm::WeaponID::WBaseballBat); }</pre>		

jun 26, 18 2:39

Still.h

Page 1/1

```

//
// Created by Gorco on 19/05/18.
//

#ifndef INC_4_WORMS_STOPMOVE_H
#define INC_4_WORMS_STOPMOVE_H

#include <Box2D/Common/b2Math.h>
#include <vector>

#include "PlayerState.h"

namespace Worms {
class Still : public State {
public:
    Still();
    ~Still() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;
};
}

#endif // INC_4_WORMS_STOPMOVE_H

```



```

jun 29, 18 16:28 Team.cpp Page 1/3
//
// Created by rodrigo on 3/06/18.
//

#include "Team.h"
#include "Weapons/AerialAttack.h"
#include "Weapons/Banana.h"
#include "Weapons/BaseballBat.h"
#include "Weapons/Bazooka.h"
#include "Weapons/Cluster.h"
#include "Weapons/Dynamite.h"
#include "Weapons/Grenade.h"
#include "Weapons/Holy.h"
#include "Weapons/Mortar.h"
#include "Weapons/Teleport.h"
#include "Weapons/WeaponNone.h"

Worms::Team::Team(std::vector<uint8_t> &playerIDs, std::vector<Player> &players,
                  const std::map<Worm::WeaponID, std::int16_t> &stageAmmo)
    : playerIDs(std::move(playerIDs)), ammunitionCounter(stageAmmo) {
    for (auto id : this->playerIDs) {
        players[id].setTeam(this);
    }
    this->initializeWeapons();
}

void Worms::Team::checkAlive(std::vector<Player> &players) {
    if (this->alive) {
        bool teamAlive = false;
        for (auto teamPlayerID : this->playerIDs) {
            if (players[teamPlayerID].getStateId() != Worm::StateID::Dead) {
                teamAlive = true;
            }
        }
        if (!teamAlive) {
            this->alive = false;
        }
    }
}

bool Worms::Team::isAlive() {
    return this->alive;
}

uint8_t Worms::Team::getCurrentPlayerID() {
    return this->playerIDs[this->currentPlayer];
}

void Worms::Team::setCurrentPlayer(uint8_t currentPlayer) {
    this->currentPlayer = currentPlayer;
}

void Worms::Team::endTurn(std::vector<Worms::Player> &players) {
    do {
        this->currentPlayer = (this->currentPlayer + 1) % this->playerIDs.size();
    } while (players[this->getCurrentPlayerID()].getStateId() == Worm::StateID::Dead);
}

std::uint32_t Worms::Team::calculateTotalHealth(std::vector<Worms::Player> &players) {

```

```

jun 29, 18 16:28 Team.cpp Page 2/3
    std::uint32_t total{0};
    for (auto playerID : this->playerIDs) {
        for (auto &player : players) {
            if (player.getId() == playerID) {
                total += (std::uint32_t)std::floor(player.health);
            }
        }
    }
    return total;
}

std::shared_ptr<Worms::Weapon> Worms::Team::getWeapon(const Worm::WeaponID &id)
{
    if (this->ammunitionCounter.at(id) == 0) {
        return this->weaponNone;
    }

    switch (id) {
        case Worm::WeaponID::WBazooka:
            return this->bazooka;
        case Worm::WeaponID::WGrenade:
            return this->grenade;
        case Worm::WeaponID::WCluster:
            return this->cluster;
        case Worm::WeaponID::WMortar:
            return this->mortar;
        case Worm::WeaponID::WBanana:
            return this->banana;
        case Worm::WeaponID::WHoly:
            return this->holy;
        case Worm::WeaponID::WAerial:
            return this->aerialAttack;
        case Worm::WeaponID::WDynamite:
            return this->dynamite;
        case Worm::WeaponID::WBaseballBat:
            return this->baseballBat;
        case Worm::WeaponID::WTeleport:
            return this->teleport;
        default:
            return this->weaponNone;
    }
}

void Worms::Team::initializeWeapons() {
    this->aerialAttack = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::AerialAttack());
    this->banana = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Banana(0.0f));
    this->baseballBat = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::BaseballBat(0.0f));
    this->bazooka = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Bazooka(0.0f));
    this->cluster = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Cluster(0.0f));
    this->dynamite = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Dynamite());
    this->grenade = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Grenade(0.0f));
    this->holy = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Holy(0.0f));
    this->mortar = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Mortar(0.0f));
    this->teleport = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::Teleport());
    this->weaponNone = std::shared_ptr<Worms::Weapon>(new Worm::Weapon::WeaponNone());
}

void Worms::Team::weaponUsed(const Worm::WeaponID weaponID) {
    if (this->ammunitionCounter.at(weaponID) > 0) {

```

jun 29, 18 16:28

Team.cpp

Page 3/3

```

        this->ammunitionCounter.at(weaponID)--;
    }
}

void Worms::Team::serialize(IO::GameStateMsg &msg) const {
    Worm::WeaponID weapons[] = {Worm::WBazooka,    Worm::WGrenade, Worm::WClust
er, Worm::WMortar,
                                Worm::WBanana,    Worm::WHoly,    Worm::WAeria
l, Worm::WDynamite,
                                Worm::WBaseballBat, Worm::WTeleport};

    for (int i = 0; i < 10; i++) {
        msg.weaponAmmunition[i] = this->ammunitionCounter.at(weapons[i]);
    }
}

void Worms::Team::kill(std::vector<Worms::Player> &players) {
    for (auto &playerID : this->playerIDs) {
        players[playerID].die();
    }
    this->alive = false;
}

```

jun 29, 18 16:28

Team.h

Page 1/1

```

//
// Created by rodrigo on 3/06/18.
//

#ifndef INC_4_WORMS_TEAM_H
#define INC_4_WORMS_TEAM_H

#include <stdint.h>
#include <cstdint>
#include <map>
#include <vector>

#include "Weapons/Weapon.h"

namespace Worms {
class Player;
class Team {
public:
    Team(std::vector<uint8_t> &playerIDs, std::vector<Player> &players,
         const std::map<Worm::WeaponID, std::int16_t> &stageAmmo);
    ~Team() = default;
    void checkAlive(std::vector<Player> &players);
    bool isAlive();
    uint8_t getCurrentPlayerID();
    void setCurrentPlayer(uint8_t currentPlayer);
    void endTurn(std::vector<Worms::Player> &players);
    std::uint32_t calculateTotalHealth(std::vector<Worms::Player> &players);
    std::shared_ptr<Weapon> getWeapon(const Worm::WeaponID &id);
    void weaponUsed(const Worm::WeaponID weaponID);
    void serialize(IO::GameStateMsg &msg) const;
    void kill(std::vector<Worms::Player> &players);

private:
    std::vector<uint8_t> playerIDs;
    uint8_t currentPlayer{0};
    bool alive{true};
    std::shared_ptr<Weapon> aerialAttack{nullptr};
    std::shared_ptr<Weapon> banana{nullptr};
    std::shared_ptr<Weapon> baseballBat{nullptr};
    std::shared_ptr<Weapon> bazooka{nullptr};
    std::shared_ptr<Weapon> cluster{nullptr};
    std::shared_ptr<Weapon> dynamite{nullptr};
    std::shared_ptr<Weapon> grenade{nullptr};
    std::shared_ptr<Weapon> holy{nullptr};
    std::shared_ptr<Weapon> mortar{nullptr};
    std::shared_ptr<Weapon> teleport{nullptr};
    std::map<Worm::WeaponID, std::int16_t> ammunitionCounter;
    std::shared_ptr<Weapon> weaponNone;

    void initializeWeapons();
};
} // namespace Worms

#endif // INC_4_WORMS_TEAM_H

```

jun 26, 18 7:40

Teleport.cpp

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#include "Teleport.h"

#define CONFIG Game::Config::getInstance()

Weapon::Teleport::Teleport()
    : Weapon::Weapon(CONFIG.getTeleportConfig(), Worm::WeaponID::WTeleport, 0.0)
{}

void Weapon::Teleport::update(float dt) {}

void Weapon::Teleport::startShot(Worms::Player *player) {}

void Weapon::Teleport::endShot() {}

void Weapon::Teleport::setTimeout(uint8_t time) {}

std::list<Worms::Bullet> Weapon::Teleport::onExplode(const Worms::Bullet &mainBullet,
                                                    Worms::Physics &physics) {
    return std::move(std::list<Worms::Bullet>());
}

void Weapon::Teleport::positionSelected(Worms::Player &p, Math::Point<float> point) {
    p.teleportPosition = point;
    p.notify(p, Event::Teleported);
    p.setState(Worm::StateID::Teleporting);
}

void Weapon::Teleport::increaseAngle() {}

void Weapon::Teleport::decreaseAngle() {}

```

jun 26, 18 7:40

Teleported.cpp

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#include "Teleported.h"
#include "../Config/Config.h"
#include "../Player.h"

Worms::Teleported::Teleported()
    : State(Worm::StateID::Teleported),
    teleportTime(Game::Config::getInstance().getTeleportTime()) {}

void Worms::Teleported::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed >= this->teleportTime) {
        p.setState(Worm::StateID::Falling);
    }
}

void Worms::Teleported::moveRight(Worms::Player &p) {}
void Worms::Teleported::moveLeft(Worms::Player &p) {}
void Worms::Teleported::jump(Worms::Player &p) {}
void Worms::Teleported::stopMove(Worms::Player &p) {}
void Worms::Teleported::backFlip(Worms::Player &p) {}
void Worms::Teleported::bazooka(Worms::Player &p) {}
void Worms::Teleported::pointUp(Worms::Player &p) {}
void Worms::Teleported::pointDown(Worms::Player &p) {}
void Worms::Teleported::startShot(Worms::Player &p) {}
void Worms::Teleported::endShot(Worms::Player &p) {}
void Worms::Teleported::grenade(Worms::Player &p) {}
void Worms::Teleported::cluster(Worms::Player &p) {}
void Worms::Teleported::mortar(Worms::Player &p) {}
void Worms::Teleported::banana(Worms::Player &p) {}
void Worms::Teleported::holy(Worms::Player &p) {}
void Worms::Teleported::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::Teleported::aerialAttack(Worms::Player &p) {}
void Worms::Teleported::dynamite(Worms::Player &p) {}
void Worms::Teleported::teleport(Worms::Player &p) {}
void Worms::Teleported::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

Teleported.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_TELEPORTED_H
#define INC_4_WORMS_TELEPORTED_H

#include <stdint-gcc.h>
#include <cstdint>
#include "PlayerState.h"

namespace Worms {
class Teleported : public State {
public:
    Teleported();
    ~Teleported() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    float teleportTime;
};
}

#endif // INC_4_WORMS_TELEPORTED_H

```

jun 26, 18 2:39

Teleport.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_TELEPORT_H
#define INC_4_WORMS_TELEPORT_H

#include "../Player.h"
#include "Weapon.h"

namespace Weapon {
class Teleport : public Worms::Weapon {
public:
    Teleport();
    ~Teleport() override = default;
    void update(float dt) override;
    void increaseAngle() override;
    void decreaseAngle() override;
    void startShot(Worms::Player *player) override;
    void endShot() override;
    void setTimeout(uint8_t time) override;
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                      Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override;
};
} // namespace Weapon

#endif // INC_4_WORMS_TELEPORT_H

```

jun 26, 18 2:39

Teleporting.cpp

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#include "Teleporting.h"
#include <Camera.h>
#include "../Config/Config.h"
#include "../Player.h"

Worms::Teleporting::Teleporting(GUI::Position p)
: State(Worm::StateID::Teleporting),
  newPosition(p),
  teleportTime(Game::Config::getInstance().getTeleportTime()) {}

void Worms::Teleporting::update(Worms::Player &p, float dt, b2Body *body) {
    this->timeElapsed += dt;
    if (this->timeElapsed >= this->teleportTime) {
        p.setPosition(this->newPosition);
        p.setState(Worm::StateID::Teleported);
    }
}

void Worms::Teleporting::moveRight(Worms::Player &p) {}
void Worms::Teleporting::moveLeft(Worms::Player &p) {}
void Worms::Teleporting::jump(Worms::Player &p) {}
void Worms::Teleporting::stopMove(Worms::Player &p) {}
void Worms::Teleporting::backFlip(Worms::Player &p) {}
void Worms::Teleporting::bazooka(Worms::Player &p) {}
void Worms::Teleporting::pointUp(Worms::Player &p) {}
void Worms::Teleporting::pointDown(Worms::Player &p) {}
void Worms::Teleporting::startShot(Worms::Player &p) {}
void Worms::Teleporting::endShot(Worms::Player &p) {}
void Worms::Teleporting::grenade(Worms::Player &p) {}
void Worms::Teleporting::cluster(Worms::Player &p) {}
void Worms::Teleporting::mortar(Worms::Player &p) {}
void Worms::Teleporting::banana(Worms::Player &p) {}
void Worms::Teleporting::holy(Worms::Player &p) {}
void Worms::Teleporting::setTimeout(Worms::Player &p, uint8_t time) {}
void Worms::Teleporting::aerialAttack(Worms::Player &p) {}
void Worms::Teleporting::dynamite(Worms::Player &p) {}
void Worms::Teleporting::teleport(Worms::Player &p) {}
void Worms::Teleporting::baseballBat(Worms::Player &p) {}

```


jun 26, 18 2:39

Teleporting.h

Page 1/1

```

//
// Created by rodrigo on 16/06/18.
//

#ifndef INC_4_WORMS_TELEPORTING_H
#define INC_4_WORMS_TELEPORTING_H

#include <Camera.h>
#include <stdint-gcc.h>
#include <cstdint>
#include "PlayerState.h"

namespace Worms {
class Teleporting : public State {
public:
    Teleporting(GUI::Position p);
    ~Teleporting() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    void pointUp(Player &p) override;
    void pointDown(Player &p) override;

private:
    float timeElapsed{0.0f};
    GUI::Position newPosition;
    float teleportTime;
};
}

#endif // INC_4_WORMS_TELEPORTING_H

```

jun 26, 18 2:39	TouchSensor.cpp	Page 1/2
-----------------	------------------------	----------

```

#include "TouchSensor.h"
#include <iostream>

/**
 * @brief Construct a TouchSensor for the given body and shape.
 *
 * @param body Body that the touch sensor belongs to.
 * @param shape Sensor shape.
 */
Worms::TouchSensor::TouchSensor(b2Body &body, b2Shape &shape) : PhysicsEntity(EntityID::Sensor) {
    /* fixture definition using the given shape */
    b2FixtureDef fixtureDef;
    fixtureDef.shape = &shape;
    fixtureDef.density = 1;
    fixtureDef.isSensor = true;

    this->fixture = body.CreateFixture(&fixtureDef);
    this->fixture->SetUserData(this);
}

Worms::TouchSensor::iterator Worms::TouchSensor::begin() {
    return this->contacts.begin();
}

Worms::TouchSensor::iterator Worms::TouchSensor::end() {
    return this->contacts.end();
}

/**
 * @brief Called whenever the sensor started contacting another entity.
 *
 * @param Contacted entity.
 */
void Worms::TouchSensor::startContact(PhysicsEntity *physicsEntity, b2Contact &contact) {
    /* checks if the entity is in the ignore list */
    if (!this->isIgnored(physicsEntity)) {
        b2Manifold manifold;
        const b2Transform t1 = contact.GetFixtureA()->GetBody()->GetTransform();
        const b2Transform t2 = contact.GetFixtureB()->GetBody()->GetTransform();

        contact.Evaluate(&manifold, t1, t2);

        if (contact.GetFixtureA()->GetUserData() == physicsEntity) {
            this->contacts[physicsEntity] = -manifold.localNormal;
        } else {
            this->contacts[physicsEntity] = manifold.localNormal;
        }
    }
}

/**
 * @brief Called whenever the sensor stopped contacting another entity.
 *
 * @param Entity.
 */
void Worms::TouchSensor::endContact(PhysicsEntity *physicsEntity, b2Contact &contact) {
    /* checks if the entity is in the ignore list */
    if (!this->isIgnored(physicsEntity)) {
        this->contacts.erase(physicsEntity);
    }
}

```

jun 26, 18 2:39	TouchSensor.cpp	Page 2/2
-----------------	------------------------	----------

```

    }
}

/**
 * @brief Whether the sensor is active or not (i.e. touching another body).
 *
 * @return true is active.
 */
bool Worms::TouchSensor::isActive() const {
    return (this->contacts.size() > 0);
}

/**
 * @brief Adds an entity that should be ignored by the sensor.
 *
 * @param other Entity to ignore.
 */
void Worms::TouchSensor::ignore(PhysicsEntity &other) {
    this->ignoredEntities.push_back(&other);
}

/**
 * @brief Checks if a given entity is in the ignore list.
 *
 * @param entity Entity to check.
 * @return true if the given entity is ignored by this sensor.
 */
bool Worms::TouchSensor::isIgnored(PhysicsEntity *entity) {
    return std::find(this->ignoredEntities.begin(), this->ignoredEntities.end(), entity) != this->ignoredEntities.end();
}

```

jun 26, 18 2:39

TouchSensor.h

Page 1/1

```

#ifndef TOUCH_SENSOR_H_
#define TOUCH_SENSOR_H_

#include <unordered_map>
#include <vector>

#include "Physics.h"
#include "PhysicsEntity.h"

namespace Worms {
class TouchSensor : public PhysicsEntity {
public:
    using iterator = std::unordered_map<PhysicsEntity *, b2Vec2>::iterator;

    TouchSensor(b2Body &body, b2Shape &shape);
    ~TouchSensor() = default;

    iterator begin();
    iterator end();

    bool isActive() const;
    void ignore(PhysicsEntity &other);

    void startContact(PhysicsEntity *physicsEntity, b2Contact &contact);
    void endContact(PhysicsEntity *physicsEntity, b2Contact &contact);

private:
    bool isIgnored(PhysicsEntity *entity);

    b2Fixture *fixture{nullptr};
    std::vector<PhysicsEntity *> ignoredEntities;
    std::unordered_map<PhysicsEntity *, b2Vec2> contacts;
    std::unordered_map<PhysicsEntity *, b2Fixture *> contactFixtures;
};
} // namespace Worms

#endif

```

jun 26, 18 2:39

Walk.cpp

Page 1/2

```

#include <cmath>
#include <iostream>
#include <memory>

#include "../Player.h"
#include "Still.h"
#include "Walk.h"

void Worms::Walk::update(Player &p, float dt, b2Body *body) {
    float32 mass = body->GetMass();
    b2Vec2 vel = body->GetLinearVelocity();

    float final_vel{0.0f};

    if (!p.isOnGround()) {
        this->impulses[0] = -vel.x * mass;
        body->ApplyLinearImpulse(b2Vec2(impulses[0], impulses[1]), body->GetWorldCenter(), true);
        p.notify(p, Event::WormFalling);
        p.setState(Worm::StateID::Falling);
        return;
    }

    if (p.direction == Worm::Direction::left) {
        final_vel = -this->walkVelocity;
    } else {
        final_vel = this->walkVelocity;
    }

    this->impulses[0] = mass * (final_vel - vel.x);
    body->ApplyLinearImpulse(b2Vec2(this->impulses[0], this->impulses[1]), body->GetWorldCenter(), true);

    p.lastWalkDirection = p.direction;

    this->timeElapsed += dt;
}

void Worms::Walk::moveRight(Worms::Player &p) {
    p.direction = Worm::Direction::right;
}

void Worms::Walk::moveLeft(Worms::Player &p) {
    p.direction = Worm::Direction::left;
}

void Worms::Walk::stopMove(Worms::Player &p) {
    p.setState(Worm::StateID::Still);
}

void Worms::Walk::jump(Worms::Player &p) {}

Worms::Walk::Walk()
    : State(Worm::StateID::Walk), walkVelocity(Game::Config::getInstance().getWalkVelocity()) {}

void Worms::Walk::backFlip(Worms::Player &p) {}

void Worms::Walk::bazooka(Worms::Player &p) {}

```

jun 26, 18 2:39

Walk.cpp

Page 2/2

```

void Worms::Walk::pointUp(Worms::Player &p) {}

void Worms::Walk::pointDown(Worms::Player &p) {}

void Worms::Walk::startShot(Worms::Player &p) {}

void Worms::Walk::endShot(Worms::Player &p) {}

void Worms::Walk::grenade(Worms::Player &p) {}

void Worms::Walk::cluster(Worms::Player &p) {}

void Worms::Walk::mortar(Worms::Player &p) {}

void Worms::Walk::banana(Worms::Player &p) {}

void Worms::Walk::holy(Worms::Player &p) {}

void Worms::Walk::setTimeout(Worms::Player &p, uint8_t time) {}

void Worms::Walk::aerialAttack(Worms::Player &p) {}

void Worms::Walk::dynamite(Worms::Player &p) {}

void Worms::Walk::teleport(Worms::Player &p) {}

void Worms::Walk::baseballBat(Worms::Player &p) {}

```

jun 26, 18 2:39

Walk.h

Page 1/1

```

#ifndef _PLAYERWALKLEFT_H
#define _PLAYERWALKLEFT_H

#include "../Config/Config.h"
#include "PlayerState.h"

namespace Worms {
class Walk : public State {
public:
    Walk();
    ~Walk() = default;
    void update(Player &p, float dt, b2Body *body) override;
    void moveRight(Player &p) override;
    void moveLeft(Player &p) override;
    void jump(Player &p) override;
    void setTimeout(Player &p, uint8_t time) override;

    void bazooka(Player &p) override;
    void grenade(Player &p) override;
    void cluster(Player &p) override;
    void mortar(Player &p) override;
    void banana(Player &p) override;
    void holy(Player &p) override;
    void aerialAttack(Player &p) override;
    void dynamite(Player &p) override;
    void baseballBat(Player &p) override;
    void teleport(Player &p) override;

    void startShot(Player &p) override;
    void endShot(Player &p) override;
    void backFlip(Player &p) override;
    void stopMove(Player &p) override;
    virtual void pointUp(Player &p) override;
    virtual void pointDown(Player &p) override;

private:
    const float walkVelocity;
    float timeElapsed{0.0f};
};
}

#endif // _PLAYERWALKLEFT_H

```

jun 26, 18 2:39

WeaponConfig.cpp

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 22/06/18
 */

#include "WeaponConfig.h"
#include "ConfigDefines.h"

Config::Weapon::Weapon(const YAML::Node &config)
: dmgInfo(config[BULLET][DAMAGE]),
  minAngle(config[ANGLE][MIN].as<float>()),
  maxAngle(config[ANGLE][MAX].as<float>()),
  angleStep(config[ANGLE][STEP].as<float>()),
  maxShotPower((std::uint16_t)config[MAX_SHOT_POWER].as<unsigned int>()),
  restitution(config[BULLET][RESTITUTION].as<float>()),
  friction(config[BULLET][FRICTION].as<float>()),
  explotionInitialTimeout(
    (std::uint8_t)config[BULLET][EXPLOSION_INITIAL_TIMEOUT].as<unsigned in
t>()),
  hasAfterExplode(config[HAS_AFTER_EXPLODE].as<bool>()),
  bulletRadius(config[BULLET][RADIUS].as<float>()),
  bulletDampingRatio(config[BULLET][DAMAGE][DAMPING_RATIO].as<float>()),
  windAffected(config[BULLET][WIND_AFFECTED].as<bool>()) {}

```

jun 26, 18 7:40

WeaponConfig.h

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 22/06/18
 */

#ifndef __WeaponConfig_H__
#define __WeaponConfig_H__

#include <stdint>
#include "yaml-cpp/node/node.h"

#include "BulletConfig.h"

namespace Config {
struct Weapon {
    Bullet::DamageInfo dmgInfo;
    float minAngle;
    float maxAngle;
    float angleStep;
    std::uint16_t maxShotPower;
    float restitution;
    float friction;
    std::uint8_t explotionInitialTimeout;
    bool hasAfterExplode;
    float bulletRadius;
    float bulletDampingRatio;
    bool windAffected;

    explicit Weapon(const YAML::Node &config);
};
} // namespace Config

#endif // __WeaponConfig_H__
```

jun 29, 18 16:28	Weapon.cpp	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 28/05/18 */ #include "Weapon.h" #include "../Config/Config.h" #include "../Player.h" #include "../Config/WeaponConfig.h" Worms::Weapon::Weapon(const Config::Weapon &config, Worm::WeaponID id, float angle) : config(config), id(id), angle(angle) { this->angle = angle; this->timeLimit = this->config.explosionInitialTimeout; /* * Because the limit angles between weapons are * different, it is necessary to check boundaries angles. * If not, the game could crash in rendering time. */ this->checkBoundaryAngles(); } const Worm::WeaponID &Worms::Weapon::getWeaponID() const { return this->id; } void Worms::Weapon::decreaseAngle() { this->angle -= this->config.angleStep; if (this->angle < this->config.minAngle) { this->angle = this->config.minAngle; } } void Worms::Weapon::increaseAngle() { this->angle += this->config.angleStep; if (this->angle > this->config.maxAngle) { this->angle = this->config.maxAngle; } } float Worms::Weapon::getAngle() const { return this->angle; } void Worms::Weapon::checkBoundaryAngles() { if (this->angle > this->config.maxAngle) { this->angle = this->config.maxAngle; } else if (this->angle < this->config.minAngle) { this->angle = this->config.minAngle; } } Worms::BulletInfo Worms::Weapon::getBulletInfo() { return Worms::BulletInfo{this->config.dmgInfo, Math::Point<float>{0, 0}, angle, this->shotPower, 0, this->config.restitution, this->config.friction, this->timeLimit, </pre>		

jun 29, 18 16:28	Weapon.cpp	Page 2/2
<pre> this->config.hasAfterExplode ? Event::OnExplode : Event::Explode, this->config.bulletRadius, this->config.bulletDampingRatio, this->config.windAffected); } void Worms::Weapon::setAngle(float angle) { this->angle = angle; } bool Worms::Weapon::isP2PWeapon() { return this->isP2P; } </pre>		

jun 29, 18 16:28	Weapon.h	Page 1/2
<pre> /* * Created by Federico Manuel Gomez Peter. * date: 28/05/18 */ #ifndef __WEAPON_H__ #define __WEAPON_H__ #include <GameStateMsg.h> #include <list> #include <memory> #include "../Config/Config.h" #include "Bullet.h" #include "../Config/WeaponConfig.h" namespace Worms { class Player; class Weapon { public: Weapon(const Config::Weapon &config, Worm::WeaponID id, float angle); virtual ~Weapon() = default; const Worm::WeaponID &getWeaponID() const; /** * If was an event of startShot, then increase its power shot until * reach its limit. * @param dt */ virtual void update(float dt) = 0; /** * @brief increases the angle of the aim. If the angle exceeds the limit * then it will be changed to the maximum possible */ virtual void increaseAngle(); /** * @brief decreases the angle of the aim. If the angle exceeds the limit * then it will be changed to the maximum possible */ virtual void decreaseAngle(); float getAngle() const; void setAngle(float angle); virtual void startShot(Worms::Player *player) = 0; virtual void endShot() = 0; BulletInfo getBulletInfo(); virtual void setTimeout(uint8_t time) = 0; /** * @brief check if the weapon is person to preson or not * @return */ bool isP2PWeapon(); /** * Used by te remote control weapons. Sends to the weapon the coordinates * of the deploy of the bullets, and a reference of Player so that the * weapons, if they are remote control. calls the appropriate method. * @param player to call deploy method (if the weapon has this feature) * @param point */ virtual void positionSelected(Worms::Player &p, Math::Point<float> point) = 0; /** * Function that returns, using move semantics, a list of bullets </pre>		

jun 29, 18 16:28	Weapon.h	Page 2/2
<pre> * depending on weapon's behavior after the main bullet explode. * @return */ virtual std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet, Worms::Physics &physics) = 0; protected: bool increaseShotPower{false}; float shotPower{0}; bool isP2P{false}; const Config::Weapon &config; Worm::WeaponID id; float angle{0}; uint8_t timeLimit; private: /** * When weapons change, their own limit angles may crash the game. * To avoid this, this function checks and correct angles between changes. */ virtual void checkBoundaryAngles(); }; } // namespace Worms #endif // __WEAPON_H__ </pre>		

jun 26, 18 7:40

WeaponNone.cpp

Page 1/1

```
/*
 * Created by Federico Manuel Gomez Peter.
 * date: 24/06/18
 */

#include "WeaponNone.h"

#define CONFIG Game::Config::getInstance()

Weapon::WeaponNone::WeaponNone()
    : Weapon::Weapon(CONFIG.getTeleportConfig(), Worm::WeaponID::WNone, 0.0) {}

std::list<Worms::Bullet> Weapon::WeaponNone::onExplode(const Worms::Bullet &main
Bullet,
                                                    Worms::Physics &physics)
{
    return std::move(std::list<Worms::Bullet>());
}
```

jun 26, 18 7:40

WeaponNone.h

Page 1/1

```

/*
 * Created by Federico Manuel Gomez Peter.
 * date: 24/06/18
 */

#ifndef __WEAPON_NONE_H__
#define __WEAPON_NONE_H__

#include "Weapon.h"

namespace Weapon {
class WeaponNone : public Worms::Weapon {
public:
    WeaponNone();
    ~WeaponNone() override = default;
    void update(float dt) override{};
    void increaseAngle() override{};
    void decreaseAngle() override{};
    void checkBoundaryAngles() override{};
    void startShot(Worms::Player *player) override{};
    void endShot() override{};
    void setTimeout(uint8_t time) override{};
    std::list<Worms::Bullet> onExplode(const Worms::Bullet &mainBullet,
                                     Worms::Physics &physics) override;
    void positionSelected(Worms::Player &p, Math::Point<float> point) override{};
};
} // namespace Weapon

#endif // __WEAPON_NONE_H__

```

jun 26, 18 2:39

WindConfig.h

Page 1/1

```
/*  
 * Created by Federico Manuel Gomez Peter.  
 * date: 22/06/18  
 */
```

```
#ifndef __WIND_CONFIG_H__  
#define __WIND_CONFIG_H__
```

```
#include "yaml-cpp/node/node.h"
```

```
namespace Config {  
struct Wind {  
    float minIntensity;  
    float maxIntensity;  
    int xDirection;  
    float instensity;  
};  
} // namespace Config
```

```
#endif //__WIND_CONFIG_H__
```

jul 01, 18 17:02	Table of Content	Page 1/5
Table of Contents		
1	AerialAttack.cpp.... sheets	1 to 1 (1) pages 1- 1 40 lines
2	AerialAttack.h..... sheets	2 to 2 (1) pages 2- 2 31 lines
3	Armory.cpp..... sheets	3 to 3 (1) pages 3- 4 65 lines
4	Armory.h..... sheets	4 to 4 (1) pages 5- 5 39 lines
5	BackFlip.cpp..... sheets	5 to 5 (1) pages 6- 7 97 lines
6	BackFlip.h..... sheets	6 to 6 (1) pages 8- 8 47 lines
7	BackgroundMusic.cpp. sheets	7 to 7 (1) pages 9- 9 32 lines
8	BackgroundMusic.h... sheets	8 to 8 (1) pages 10- 10 29 lines
9	BackgroundMusicManager.h sheets	9 to 9 (1) pages 11- 11 59 lines
10	BackgroundMusicPlayer.cpp sheets	10 to 10 (1) pages 12- 12 15 lines
11	BackgroundMusicPlayer.h sheets	11 to 11 (1) pages 13- 13 29 lines
12	Banana.cpp..... sheets	12 to 12 (1) pages 14- 14 44 lines
13	Banana.h..... sheets	13 to 13 (1) pages 15- 15 36 lines
14	BaseballBat.cpp..... sheets	14 to 14 (1) pages 16- 16 36 lines
15	BaseballBat.h..... sheets	15 to 15 (1) pages 17- 17 31 lines
16	Batting.cpp..... sheets	16 to 16 (1) pages 18- 19 96 lines
17	Batting.h..... sheets	17 to 17 (1) pages 20- 21 66 lines
18	Bazooka.cpp..... sheets	18 to 18 (1) pages 22- 22 44 lines
19	Bazooka.h..... sheets	19 to 19 (1) pages 23- 23 36 lines
20	Bullet.cpp..... sheets	20 to 21 (2) pages 24- 26 122 lines
21	Bullet.h..... sheets	22 to 22 (1) pages 27- 27 51 lines
22	Button.cpp..... sheets	23 to 23 (1) pages 28- 29 70 lines
23	Button.h..... sheets	24 to 24 (1) pages 30- 30 38 lines
24	ClientSocket.cpp.... sheets	25 to 25 (1) pages 31- 31 50 lines
25	ClientSocket.h..... sheets	26 to 26 (1) pages 32- 32 23 lines
26	Cluster.cpp..... sheets	27 to 27 (1) pages 33- 33 44 lines
27	Cluster.h..... sheets	28 to 28 (1) pages 34- 34 36 lines
28	CommunicationProtocol.cpp sheets	29 to 30 (2) pages 35- 37 133 lines
29	CommunicationProtocol.h sheets	31 to 31 (1) pages 38- 38 57 lines
30	ConnectionWindow.cpp sheets	32 to 32 (1) pages 39- 40 94 lines
31	ConnectionWindow.h.. sheets	33 to 33 (1) pages 41- 41 48 lines
32	CreateGameWindow.cpp sheets	34 to 34 (1) pages 42- 43 96 lines
33	CreateGameWindow.h.. sheets	35 to 35 (1) pages 44- 44 45 lines
34	Dead.cpp..... sheets	36 to 36 (1) pages 45- 46 100 lines
35	Dead.h..... sheets	37 to 37 (1) pages 47- 47 46 lines
36	Die.cpp..... sheets	38 to 38 (1) pages 48- 49 97 lines
37	Die.h..... sheets	39 to 39 (1) pages 50- 50 46 lines
38	Drowning.cpp..... sheets	40 to 40 (1) pages 51- 52 97 lines
39	Drowning.h..... sheets	41 to 41 (1) pages 53- 53 46 lines
40	Dynamite.cpp..... sheets	42 to 42 (1) pages 54- 54 28 lines
41	Dynamite.h..... sheets	43 to 43 (1) pages 55- 55 28 lines
42	Explosion.cpp..... sheets	44 to 44 (1) pages 56- 56 28 lines
43	Explosion.h..... sheets	45 to 45 (1) pages 57- 57 30 lines
44	Falling.cpp..... sheets	46 to 46 (1) pages 58- 59 96 lines
45	Falling.h..... sheets	47 to 47 (1) pages 60- 60 46 lines
46	GameBackgroundMusic.h sheets	48 to 48 (1) pages 61- 61 23 lines
47	GameEndWindow.cpp... sheets	49 to 49 (1) pages 62- 62 54 lines
48	GameEndWindow.h.... sheets	50 to 50 (1) pages 63- 63 38 lines
49	GameSoundEffects.h.. sheets	51 to 51 (1) pages 64- 64 35 lines
50	GameTextures.h..... sheets	52 to 52 (1) pages 65- 66 74 lines
51	GameWindow.cpp..... sheets	53 to 53 (1) pages 67- 67 14 lines
52	GameWindow.h..... sheets	54 to 54 (1) pages 68- 69 83 lines
53	Grenade.cpp..... sheets	55 to 55 (1) pages 70- 70 44 lines
54	Grenade.h..... sheets	56 to 56 (1) pages 71- 71 38 lines
55	GUIGame.cpp..... sheets	57 to 62 (6) pages 72- 83 650 lines
56	GUIGame.h..... sheets	63 to 63 (1) pages 84- 85 94 lines
57	Hit.cpp..... sheets	64 to 64 (1) pages 86- 87 97 lines
58	Hit.h..... sheets	65 to 65 (1) pages 88- 88 46 lines
59	Holy.cpp..... sheets	66 to 66 (1) pages 89- 89 44 lines
60	Holy.h..... sheets	67 to 67 (1) pages 90- 90 36 lines
61	JoinGameWindow.cpp.. sheets	68 to 68 (1) pages 91- 92 105 lines

jul 01, 18 17:02	Table of Content	Page 2/5
62	JoinGameWindow.h.... sheets	69 to 69 (1) pages 93- 93 34 lines
63	Land.cpp..... sheets	70 to 70 (1) pages 94- 95 96 lines
64	Land.h..... sheets	71 to 71 (1) pages 96- 96 45 lines
65	LobbyAssistant.cpp.. sheets	72 to 73 (2) pages 97-100 195 lines
66	LobbyAssistant.h.... sheets	74 to 74 (1) pages 101-101 58 lines
67	main.cpp..... sheets	75 to 75 (1) pages 102-102 49 lines
68	Mortar.cpp..... sheets	76 to 76 (1) pages 103-103 44 lines
69	Mortar.h..... sheets	77 to 77 (1) pages 104-104 36 lines
70	PowerBar.cpp..... sheets	78 to 78 (1) pages 105-105 48 lines
71	PowerBar.h..... sheets	79 to 79 (1) pages 106-106 40 lines
72	Scope.cpp..... sheets	80 to 80 (1) pages 107-107 27 lines
73	Scope.h..... sheets	81 to 81 (1) pages 108-108 30 lines
74	SelectActionWindow.cpp sheets	82 to 82 (1) pages 109-109 55 lines
75	SelectActionWindow.h sheets	83 to 83 (1) pages 110-110 34 lines
76	Sliding.cpp..... sheets	84 to 84 (1) pages 111-112 92 lines
77	Sliding.h..... sheets	85 to 85 (1) pages 113-113 41 lines
78	SoundEffect.cpp.... sheets	86 to 86 (1) pages 114-114 32 lines
79	SoundEffect.h..... sheets	87 to 87 (1) pages 115-115 27 lines
80	SoundEffectManager.h sheets	88 to 88 (1) pages 116-116 59 lines
81	SoundEffectPlayer.cpp sheets	89 to 89 (1) pages 117-117 33 lines
82	SoundEffectPlayer.h. sheets	90 to 90 (1) pages 118-118 33 lines
83	Teleport.cpp..... sheets	91 to 91 (1) pages 119-119 39 lines
84	Teleported.cpp..... sheets	92 to 92 (1) pages 120-121 96 lines
85	Teleported.h..... sheets	93 to 93 (1) pages 122-122 45 lines
86	Teleport.h..... sheets	94 to 94 (1) pages 123-123 30 lines
87	Teleporting.cpp.... sheets	95 to 95 (1) pages 124-125 96 lines
88	Teleporting.h..... sheets	96 to 96 (1) pages 126-126 45 lines
89	WaitingPlayersWindow.cpp sheets	97 to 97 (1) pages 127-127 51 lines
90	WaitingPlayersWindow.h sheets	98 to 98 (1) pages 128-128 39 lines
91	Water.cpp..... sheets	99 to 99 (1) pages 129-129 27 lines
92	Water.h..... sheets	100 to 100 (1) pages 130-130 24 lines
93	Weapon.cpp..... sheets	101 to 101 (1) pages 131-131 21 lines
94	Weapon.h..... sheets	102 to 102 (1) pages 132-133 67 lines
95	WeaponNone.cpp..... sheets	103 to 103 (1) pages 134-134 24 lines
96	WeaponNone.h..... sheets	104 to 104 (1) pages 135-135 28 lines
97	Wind.cpp..... sheets	105 to 105 (1) pages 136-136 20 lines
98	Wind.h..... sheets	106 to 106 (1) pages 137-137 30 lines
99	WormBackFlipping.cpp sheets	107 to 107 (1) pages 138-139 97 lines
100	WormBackFlipping.h.. sheets	108 to 108 (1) pages 140-140 47 lines
101	Worm.cpp..... sheets	109 to 113 (5) pages 141-150 533 lines
102	WormEndBackFlip.cpp. sheets	114 to 114 (1) pages 151-152 97 lines
103	WormEndBackFlip.h... sheets	115 to 115 (1) pages 153-153 47 lines
104	WormEndJump.cpp..... sheets	116 to 116 (1) pages 154-155 97 lines
105	WormEndJump.h..... sheets	117 to 117 (1) pages 156-156 47 lines
106	Worm.h..... sheets	118 to 119 (2) pages 157-159 125 lines
107	WormJumping.cpp..... sheets	120 to 120 (1) pages 160-161 97 lines
108	WormJumping.h..... sheets	121 to 121 (1) pages 162-162 47 lines
109	WormStartJump.cpp... sheets	122 to 122 (1) pages 163-164 97 lines
110	WormStartJump.h..... sheets	123 to 123 (1) pages 165-165 48 lines
111	WormState.h..... sheets	124 to 124 (1) pages 166-166 61 lines
112	WormStill.cpp..... sheets	125 to 125 (1) pages 167-168 114 lines
113	WormStill.h..... sheets	126 to 126 (1) pages 169-169 48 lines
114	WormWalk.cpp..... sheets	127 to 127 (1) pages 170-171 104 lines
115	WormWalk.h..... sheets	128 to 128 (1) pages 172-172 50 lines
116	editor.cpp..... sheets	129 to 129 (1) pages 173-173 43 lines
117	editor.h..... sheets	130 to 130 (1) pages 174-174 21 lines
118	Editor.pro..... sheets	131 to 132 (2) pages 175-177 134 lines
119	editorscene.cpp..... sheets	133 to 134 (2) pages 178-180 134 lines
120	editorscene.h..... sheets	135 to 135 (1) pages 181-181 55 lines
121	editorview.cpp..... sheets	136 to 137 (2) pages 182-184 174 lines
122	editorview.h..... sheets	138 to 138 (1) pages 185-185 49 lines
123	main.cpp..... sheets	139 to 139 (1) pages 186-186 11 lines

jul 01, 18 17:02	Table of Content	Page 3/5
124	<i>mainwindow.cpp</i> sheets 140 to 141 (2) pages 187-189	124 lines
125	<i>mainwindow.h</i> sheets 142 to 142 (1) pages 190-190	43 lines
126	<i>mainwindow.ui</i> sheets 143 to 148 (6) pages 191-201	626 lines
127	<i>qgraphicsitemlayer.cpp</i> sheets 149 to 149 (1) pages 202-202	10 lines
128	<i>qgraphicsitemlayer.h</i> sheets 150 to 150 (1) pages 203-203	16 lines
129	<i>resources.qrc</i> sheets 151 to 151 (1) pages 204-204	11 lines
130	<i>stagedata.cpp</i> sheets 152 to 153 (2) pages 205-207	127 lines
131	<i>stagedata.h</i> sheets 154 to 154 (1) pages 208-208	55 lines
132	<i>stageelement.cpp</i> sheets 155 to 155 (1) pages 209-210	65 lines
133	<i>stageelement.h</i> sheets 156 to 156 (1) pages 211-211	47 lines
134	<i>stageelementworm.cpp</i> sheets 157 to 157 (1) pages 212-212	20 lines
135	<i>stageelementworm.h</i> .. sheets 158 to 158 (1) pages 213-213	18 lines
136	<i>stageelemllonggirder.cpp</i> sheets 159 to 159 (1) pages 214-214	20 lines
137	<i>stageelemllonggirder.h</i> sheets 160 to 160 (1) pages 215-215	16 lines
138	<i>stageelemshortgirder.cpp</i> sheets 161 to 161 (1) pages 216-216	20 lines
139	<i>stageelemshortgirder.h</i> sheets 162 to 162 (1) pages 217-217	16 lines
140	<i>Animation.cpp</i> sheets 163 to 164 (2) pages 218-220	128 lines
141	<i>Animation.h</i> sheets 165 to 165 (1) pages 221-222	71 lines
142	<i>Buffer.cpp</i> sheets 166 to 166 (1) pages 223-224	67 lines
143	<i>Buffer.h</i> sheets 167 to 167 (1) pages 225-226	91 lines
144	<i>Camera.cpp</i> sheets 168 to 170 (3) pages 227-232	300 lines
145	<i>Camera.h</i> sheets 171 to 171 (1) pages 233-234	64 lines
146	<i>Chronometer.cpp</i> sheets 172 to 172 (1) pages 235-235	22 lines
147	<i>Chronometer.h</i> sheets 173 to 173 (1) pages 236-236	15 lines
148	<i>Color.h</i> sheets 174 to 174 (1) pages 237-237	17 lines
149	<i>CommunicationSocket.cpp</i> sheets 175 to 175 (1) pages 238-239	64 lines
150	<i>CommunicationSocket.h</i> sheets 176 to 176 (1) pages 240-240	43 lines
151	<i>Direction.h</i> sheets 177 to 177 (1) pages 241-241	14 lines
152	<i>DoubleBuffer.h</i> sheets 178 to 178 (1) pages 242-243	92 lines
153	<i>ErrorMessages.h</i> sheets 179 to 179 (1) pages 244-244	25 lines
154	<i>Exception.cpp</i> sheets 180 to 180 (1) pages 245-245	39 lines
155	<i>Exception.h</i> sheets 181 to 181 (1) pages 246-246	29 lines
156	<i>Font.cpp</i> sheets 182 to 182 (1) pages 247-247	21 lines
157	<i>Font.h</i> sheets 183 to 183 (1) pages 248-248	24 lines
158	<i>GameStateMsg.h</i> sheets 184 to 186 (3) pages 249-254	325 lines
159	<i>IO.h</i> sheets 187 to 187 (1) pages 255-255	18 lines
160	<i>Observer.h</i> sheets 188 to 188 (1) pages 256-256	50 lines
161	<i>Point.h</i> sheets 189 to 189 (1) pages 257-258	77 lines
162	<i>Protocol.h</i> sheets 190 to 191 (2) pages 259-262	208 lines
163	<i>Socket.cpp</i> sheets 192 to 192 (1) pages 263-263	47 lines
164	<i>Socket.h</i> sheets 193 to 193 (1) pages 264-264	35 lines
165	<i>Stage.cpp</i> sheets 194 to 195 (2) pages 265-268	194 lines
166	<i>Stage.h</i> sheets 196 to 196 (1) pages 269-270	69 lines
167	<i>Stream.h</i> sheets 197 to 197 (1) pages 271-272	78 lines
168	<i>Subject.cpp</i> sheets 198 to 198 (1) pages 273-273	21 lines
169	<i>Subject.h</i> sheets 199 to 199 (1) pages 274-274	34 lines
170	<i>Text.cpp</i> sheets 200 to 200 (1) pages 275-276	107 lines
171	<i>Text.h</i> sheets 201 to 201 (1) pages 277-277	59 lines
172	<i>Texture.cpp</i> sheets 202 to 202 (1) pages 278-279	79 lines
173	<i>Texture.h</i> sheets 203 to 203 (1) pages 280-280	27 lines
174	<i>TextureManager.h</i> sheets 204 to 204 (1) pages 281-281	57 lines
175	<i>Thread.cpp</i> sheets 205 to 205 (1) pages 282-282	24 lines
176	<i>Thread.h</i> sheets 206 to 206 (1) pages 283-283	61 lines
177	<i>utils.h</i> sheets 207 to 207 (1) pages 284-284	19 lines
178	<i>Window.cpp</i> sheets 208 to 209 (2) pages 285-287	133 lines
179	<i>Window.h</i> sheets 210 to 210 (1) pages 288-288	39 lines
180	<i>WrapTexture.cpp</i> sheets 211 to 211 (1) pages 289-290	96 lines
181	<i>WrapTexture.h</i> sheets 212 to 212 (1) pages 291-291	24 lines
182	<i>AerialAttack.cpp</i> sheets 213 to 213 (1) pages 292-292	53 lines
183	<i>AerialAttack.h</i> sheets 214 to 214 (1) pages 293-293	32 lines
184	<i>BackFlipping.cpp</i> sheets 215 to 215 (1) pages 294-295	75 lines
185	<i>BackFlipping.h</i> sheets 216 to 216 (1) pages 296-296	49 lines

jul 01, 18 17:02	Table of Content	Page 4/5
186	<i>Banana.cpp</i> sheets 217 to 217 (1) pages 297-297	41 lines
187	<i>Banana.h</i> sheets 218 to 218 (1) pages 298-298	30 lines
188	<i>BaseballBat.cpp</i> sheets 219 to 219 (1) pages 299-299	38 lines
189	<i>BaseballBat.h</i> sheets 220 to 220 (1) pages 300-300	32 lines
190	<i>Batting.cpp</i> sheets 221 to 221 (1) pages 301-301	58 lines
191	<i>Batting.h</i> sheets 222 to 222 (1) pages 302-302	46 lines
192	<i>Bazooka.cpp</i> sheets 223 to 223 (1) pages 303-303	43 lines
193	<i>Bazooka.h</i> sheets 224 to 224 (1) pages 304-304	31 lines
194	<i>BulletConfig.cpp</i> sheets 225 to 225 (1) pages 305-305	13 lines
195	<i>BulletConfig.h</i> sheets 226 to 226 (1) pages 306-306	25 lines
196	<i>Bullet.cpp</i> sheets 227 to 228 (2) pages 307-309	125 lines
197	<i>Bullet.h</i> sheets 229 to 229 (1) pages 310-311	88 lines
198	<i>Cluster.cpp</i> sheets 230 to 230 (1) pages 312-313	64 lines
199	<i>Cluster.h</i> sheets 231 to 231 (1) pages 314-314	33 lines
200	<i>Config.cpp</i> sheets 232 to 233 (2) pages 315-318	210 lines
201	<i>ConfigDefines.h</i> sheets 234 to 234 (1) pages 319-320	64 lines
202	<i>Config.h</i> sheets 235 to 236 (2) pages 321-323	141 lines
203	<i>ContactEventListener.cpp</i> sheets 237 to 237 (1) pages 324-325	90 lines
204	<i>ContactEventListener.h</i> sheets 238 to 238 (1) pages 326-326	22 lines
205	<i>Dead.cpp</i> sheets 239 to 239 (1) pages 327-327	52 lines
206	<i>Dead.h</i> sheets 240 to 240 (1) pages 328-328	44 lines
207	<i>Die.cpp</i> sheets 241 to 241 (1) pages 329-329	62 lines
208	<i>Die.h</i> sheets 242 to 242 (1) pages 330-330	49 lines
209	<i>Drowning.cpp</i> sheets 243 to 243 (1) pages 331-331	59 lines
210	<i>Drowning.h</i> sheets 244 to 244 (1) pages 332-332	49 lines
211	<i>Dynamite.cpp</i> sheets 245 to 245 (1) pages 333-333	34 lines
212	<i>Dynamite.h</i> sheets 246 to 246 (1) pages 334-334	29 lines
213	<i>EndBackFlip.cpp</i> sheets 247 to 247 (1) pages 335-335	60 lines
214	<i>EndBackFlip.h</i> sheets 248 to 248 (1) pages 336-336	48 lines
215	<i>EndJump.cpp</i> sheets 249 to 249 (1) pages 337-337	59 lines
216	<i>EndJump.h</i> sheets 250 to 250 (1) pages 338-338	47 lines
217	<i>Falling.cpp</i> sheets 251 to 251 (1) pages 339-339	55 lines
218	<i>Falling.h</i> sheets 252 to 252 (1) pages 340-340	46 lines
219	<i>GameClock.cpp</i> sheets 253 to 253 (1) pages 341-341	53 lines
220	<i>GameClock.h</i> sheets 254 to 254 (1) pages 342-342	33 lines
221	<i>Game.cpp</i> sheets 255 to 259 (5) pages 343-352	573 lines
222	<i>Game.h</i> sheets 260 to 260 (1) pages 353-354	109 lines
223	<i>GameLobbyAssistant.cpp</i> sheets 261 to 262 (2) pages 355-357	137 lines
224	<i>GameLobbyAssistant.h</i> sheets 263 to 263 (1) pages 358-358	49 lines
225	<i>GameLobby.cpp</i> sheets 264 to 265 (2) pages 359-362	225 lines
226	<i>GameLobby.h</i> sheets 266 to 266 (1) pages 363-363	53 lines
227	<i>GamesGetter.cpp</i> sheets 267 to 267 (1) pages 364-364	19 lines
228	<i>GamesGetter.h</i> sheets 268 to 268 (1) pages 365-365	23 lines
229	<i>GameTeams.cpp</i> sheets 269 to 270 (2) pages 366-368	122 lines
230	<i>GameTeams.h</i> sheets 271 to 271 (1) pages 369-369	40 lines
231	<i>GameTurn.cpp</i> sheets 272 to 272 (1) pages 370-371	112 lines
232	<i>GameTurn.h</i> sheets 273 to 273 (1) pages 372-372	46 lines
233	<i>GameTurnState.cpp</i> .. sheets 274 to 274 (1) pages 373-373	32 lines
234	<i>GameTurnState.h</i> sheets 275 to 275 (1) pages 374-374	46 lines
235	<i>Girder.cpp</i> sheets 276 to 276 (1) pages 375-375	29 lines
236	<i>Girder.h</i> sheets 277 to 277 (1) pages 376-376	21 lines
237	<i>Grenade.cpp</i> sheets 278 to 278 (1) pages 377-377	42 lines
238	<i>Grenade.h</i> sheets 279 to 279 (1) pages 378-378	30 lines
239	<i>Hit.cpp</i> sheets 280 to 280 (1) pages 379-380	76 lines
240	<i>Hit.h</i> sheets 281 to 281 (1) pages 381-381	47 lines
241	<i>Holy.cpp</i> sheets 282 to 282 (1) pages 382-382	41 lines
242	<i>Holy.h</i> sheets 283 to 283 (1) pages 383-383	30 lines
243	<i>ImpactOnCourse.cpp</i> .. sheets 284 to 284 (1) pages 384-385	103 lines
244	<i>ImpactOnCourse.h</i> sheets 285 to 285 (1) pages 386-386	42 lines
245	<i>Jumping.cpp</i> sheets 286 to 286 (1) pages 387-388	81 lines
246	<i>Jumping.h</i> sheets 287 to 287 (1) pages 389-389	50 lines
247	<i>Land.cpp</i> sheets 288 to 288 (1) pages 390-391	65 lines

jul 01, 18 17:02		Table of Content					Page 5/5	
248	Land.h.....	sheets	289 to 289	(1)	pages 392-392	47	lines	
249	Lobbies.cpp.....	sheets	290 to 290	(1)	pages 393-394	60	lines	
250	Lobbies.h.....	sheets	291 to 291	(1)	pages 395-395	40	lines	
251	Lobby.cpp.....	sheets	292 to 292	(1)	pages 396-397	121	lines	
252	Lobby.h.....	sheets	293 to 293	(1)	pages 398-398	57	lines	
253	LobbyJoiner.cpp.....	sheets	294 to 294	(1)	pages 399-400	69	lines	
254	LobbyJoiner.h.....	sheets	295 to 295	(1)	pages 401-401	35	lines	
255	main.cpp.....	sheets	296 to 296	(1)	pages 402-403	76	lines	
256	Mortar.cpp.....	sheets	297 to 297	(1)	pages 404-405	61	lines	
257	Mortar.h.....	sheets	298 to 298	(1)	pages 406-406	30	lines	
258	P2PWeapon.cpp.....	sheets	299 to 299	(1)	pages 407-407	7	lines	
259	P2PWeapon.h.....	sheets	300 to 300	(1)	pages 408-408	24	lines	
260	Physics.cpp.....	sheets	301 to 301	(1)	pages 409-409	40	lines	
261	PhysicsEntity.cpp...	sheets	302 to 302	(1)	pages 410-410	20	lines	
262	PhysicsEntity.h.....	sheets	303 to 303	(1)	pages 411-411	35	lines	
263	Physics.h.....	sheets	304 to 304	(1)	pages 412-412	34	lines	
264	Player.cpp.....	sheets	305 to 310	(6)	pages 413-423	592	lines	
265	Player.h.....	sheets	311 to 312	(2)	pages 424-426	154	lines	
266	PlayerShot.cpp.....	sheets	313 to 313	(1)	pages 427-427	26	lines	
267	PlayerShot.h.....	sheets	314 to 314	(1)	pages 428-428	30	lines	
268	PlayerState.cpp.....	sheets	315 to 315	(1)	pages 429-429	14	lines	
269	PlayerState.h.....	sheets	316 to 316	(1)	pages 430-430	48	lines	
270	ServerSocket.cpp....	sheets	317 to 317	(1)	pages 431-432	93	lines	
271	ServerSocket.h.....	sheets	318 to 318	(1)	pages 433-433	26	lines	
272	Sliding.cpp.....	sheets	319 to 319	(1)	pages 434-435	76	lines	
273	Sliding.h.....	sheets	320 to 320	(1)	pages 436-436	42	lines	
274	StartBackFlip.cpp...	sheets	321 to 321	(1)	pages 437-438	78	lines	
275	StartBackFlip.h.....	sheets	322 to 322	(1)	pages 439-439	52	lines	
276	StartJump.cpp.....	sheets	323 to 323	(1)	pages 440-441	80	lines	
277	StartJump.h.....	sheets	324 to 324	(1)	pages 442-442	51	lines	
278	StartTurn.cpp.....	sheets	325 to 325	(1)	pages 443-443	53	lines	
279	StartTurn.h.....	sheets	326 to 326	(1)	pages 444-444	30	lines	
280	Still.cpp.....	sheets	327 to 327	(1)	pages 445-446	104	lines	
281	Still.h.....	sheets	328 to 328	(1)	pages 447-447	45	lines	
282	Team.cpp.....	sheets	329 to 330	(2)	pages 448-450	138	lines	
283	Team.h.....	sheets	331 to 331	(1)	pages 451-451	55	lines	
284	Teleport.cpp.....	sheets	332 to 332	(1)	pages 452-452	34	lines	
285	Teleported.cpp.....	sheets	333 to 333	(1)	pages 453-453	59	lines	
286	Teleported.h.....	sheets	334 to 334	(1)	pages 454-454	48	lines	
287	Teleport.h.....	sheets	335 to 335	(1)	pages 455-455	29	lines	
288	Teleporting.cpp.....	sheets	336 to 336	(1)	pages 456-456	62	lines	
289	Teleporting.h.....	sheets	337 to 337	(1)	pages 457-457	50	lines	
290	TouchSensor.cpp.....	sheets	338 to 338	(1)	pages 458-459	91	lines	
291	TouchSensor.h.....	sheets	339 to 339	(1)	pages 460-460	38	lines	
292	Walk.cpp.....	sheets	340 to 340	(1)	pages 461-462	87	lines	
293	Walk.h.....	sheets	341 to 341	(1)	pages 463-463	43	lines	
294	WeaponConfig.cpp....	sheets	342 to 342	(1)	pages 464-464	23	lines	
295	WeaponConfig.h.....	sheets	343 to 343	(1)	pages 465-465	34	lines	
296	Weapon.cpp.....	sheets	344 to 344	(1)	pages 466-467	75	lines	
297	Weapon.h.....	sheets	345 to 345	(1)	pages 468-469	87	lines	
298	WeaponNone.cpp.....	sheets	346 to 346	(1)	pages 470-470	17	lines	
299	WeaponNone.h.....	sheets	347 to 347	(1)	pages 471-471	30	lines	
300	WindConfig.h.....	sheets	348 to 348	(1)	pages 472-472	21	lines	