

jul 21, 20 15:47

tests.cpp

Page 1/1

```

1  //
2  // Created by marcos on 22/6/20.
3  //
4
5  #define CATCH_CONFIG_MAIN
6  #include "catch.hpp"
7

```

jul 21, 20 15:47

MapTests.h

Page 1/1

```

1  //
2  // Created by agustin on 22/6/20.
3  //
4
5  #ifndef ARGENTUM_MAPTESTS_H
6  #define ARGENTUM_MAPTESTS_H
7
8  class Map;
9
10 class MapTests {
11 private:
12     static void _fillEmptyMap(Map& map, int iSize, int jSize, bool isCity = false);
13
14 public:
15     static bool testAvailableMapHasAvailableTiles();
16     static bool testCityMapHasCityTiles();
17     static bool testMixedCityAndUnavailableTiles();
18     static bool testAddedItemsToMap();
19     static bool testAddedGoldToMap();
20     static bool testAddedMultipleGoldsToMapWithList();
21     static bool testAddedMultipleGoldsToMapWithoutList();
22     static bool testAddedMultipleItemsListsToMap();
23     static bool testGetTargetsOnEmptyMapReturnsEmptyList();
24     static bool testGetTargetsOnMapWithPlayerReturnsListWithOneElement();
25     static bool testGetTargetsOnMapWithDeadPlayerReturnsEmptyList();
26     static bool testPositionWithPlayerIsOccupied();
27     static bool testPositionWithMonsterIsOccupied();
28
29     static bool testListOnEmptyTileReturnsEmptyList();
30     static bool testListOnEmptyMapReturnsEmptyList();
31     static bool testWithdrawOnEmptyTileGetsNoItem();
32     static bool testWithdrawOnEmptyMapGetsNoItem();
33     static bool testDepositExistentItemOnEmptyTileGetsNoItem();
34     static bool testDepositExistentItemOnEmptyMapGetsNoItem();
35     static bool testBuyItemFromEmptyTileGetsNoItem();
36     static bool testBuyItemFromEmptyMapGetsNoItem();
37     static bool testSellExistentItemToEmptyTileGetsNoItem();
38     static bool testSellExistentItemToEmptyMapGetsNoItem();
39
40     static bool testMoveEntity();
41     static bool testRemoveEntityOnEmptyTileLeavesNoEntity();
42 };
43
44
45 #endif // ARGENTUM_MAPTESTS_H

```

jul 21, 20 15:47

MapTests.cpp

Page 1/9

```

1  //
2  // Created by agustin on 22/6/20.
3  //
4
5  #include "MapTests.h"
6
7  #include <iostream>
8  #include <memory>
9  #include "../Map/Map.h"
10 #include "../Items/Miscellaneous/Gold.h"
11 #include "../Items/Miscellaneous/HealthPotion.h"
12 #include "../Items/Miscellaneous/ManaPotion.h"
13 #include "../Items/Defense/Head.h"
14 #include "../Items/Defense/Shield.h"
15 #include "../Items/Defense/Chest.h"
16 #include "../Items/Attack/Weapon.h"
17 #include "../Entities/Player.h"
18 #include "../Entities/Monster.h"
19 #include "../Game/Game.h"
20 #include "../Entities/Citizens/Storage.h"
21 #include "catch.hpp"
22 #include "fakeit.hpp"
23
24 using namespace fakeit;
25
26 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
27
28 void MapTests::_fillEmptyMap(Map &map, int iSize, int jSize, bool isCity) {
29     for (int i = 0; i < iSize; ++i) {
30         map.tiles.emplace_back();
31         for (int j = 0; j < jSize; ++j) {
32             map.tiles[i].emplace_back(true, isCity, GameType::FloorType::GRASS0,
33                                     GameType::Structure::NO_STRUCTURE,
34                                     std::shared_ptr<Entity>(nullptr));
35         }
36     }
37 }
38
39 ///////////////////////////////////////////////////PUBLIC////////////////////////////////////
40
41 bool MapTests::testAvailableMapHasAvailableTiles() {
42     Map map;
43
44     int mapXSize = 50;
45     int mapYSize = 50;
46     _fillEmptyMap(map, mapXSize, mapYSize);
47     for (int i = 0; i < mapXSize; ++i) {
48         for (int j = 0; j < mapYSize; ++j) {
49             if (!map.tiles[i][j].isAvailable()) {
50                 return false;
51             }
52         }
53     }
54     return true;
55 }
56
57 bool MapTests::testCityMapHasCityTiles() {
58     Map map;
59     int mapXSize = 50;
60     int mapYSize = 50;
61     _fillEmptyMap(map, mapXSize, mapYSize, true);
62     for (int i = 0; i < mapXSize; ++i) {
63         for (int j = 0; j < mapYSize; ++j) {
64             if (!map.tiles[i][j].isInCity()) {
65                 return false;
66             }
67         }
68     }

```

jul 21, 20 15:47

MapTests.cpp

Page 2/9

```

67     }
68 }
69 return true;
70 }
71
72 bool MapTests::testMixedCityAndUnavailableTiles() {
73     Map map;
74     int mapXSize = 50;
75     int mapYSize = 50;
76     bool isCity;
77     bool isAvailable;
78     bool isOccupable;
79     for (int i = 0; i < mapXSize; ++i) {
80         map.tiles.emplace_back();
81         for (int j = 0; j < mapYSize; ++j) {
82             isCity = (j % 2 == 0);
83             isOccupable = j % 3 == 0;
84             map.tiles[i].emplace_back(isOccupable, isCity, GameType::FloorType::
85                                     GRASS0,
86                                     GameType::Structure::NO_STRUCTURE,
87                                     nullptr);
88         }
89     }
90     for (int i = 0; i < mapXSize; ++i) {
91         for (int j = 0; j < mapYSize; ++j) {
92             isCity = map.tiles[i][j].isInCity();
93             isAvailable = map.tiles[i][j].isAvailable();
94             if (j % 2 == 0) {
95                 if (!isCity) {
96                     return false;
97                 }
98             } else {
99                 if (isCity) {
100                     return false;
101                 }
102             }
103             if (j % 3 == 0) {
104                 if (!isAvailable) {
105                     return false;
106                 }
107             } else {
108                 if (isAvailable) {
109                     return false;
110                 }
111             }
112         }
113     }
114     return true;
115 }
116
117 bool MapTests::testAddedItemsToMap() {
118     Map map;
119     int mapXSize = 50;
120     int mapYSize = 50;
121     _fillEmptyMap(map, mapXSize, mapYSize);
122     std::list<std::shared_ptr<Item>> items;
123     items.emplace_back(new Gold(5));
124     items.emplace_back(new HealthPotion());
125     items.emplace_back(new ManaPotion());
126     items.emplace_back(new Head(GameType::Clothing::MAGIC_HAT));
127     items.emplace_back(new Head(GameType::Clothing::NO_HELMET));
128     items.emplace_back(new Shield(GameType::Clothing::IRON_SHIELD));
129     items.emplace_back(new Chest(GameType::Clothing::PLATE_ARMOR));
130     items.emplace_back(new Chest(GameType::Clothing::COMMON_CLOTHING));
131     items.emplace_back(new Weapon(GameType::Weapon::LONGSWORD));
132     items.emplace_back(new Weapon(GameType::Weapon::FIST));

```

jul 21, 20 15:47

MapTests.cpp

Page 3/9

```

132 Configuration& config = Configuration::getInstance();
133
134 std::vector<std::string> itemsNames = {config.configGetGoldName(), config.co
nfigPotionData(GameType::Potion::HEALTH_POTION).name,
135
136 config.configPotionData(GameType::Potio
n::MANA_POTION).name,
137
138 config.configClothingData(GameType::Clo
thing::MAGIC_HAT).name,
139
140 config.configClothingData(GameType::Clo
thing::NO_HELMET).name,
141
142 config.configClothingData(GameType::Clo
thing::IRON_SHIELD).name,
143
144 config.configClothingData(GameType::Clo
thing::PLATE_ARMOR).name,
145
146 config.configClothingData(GameType::Clo
thing::COMMON_CLOTHING).name,
147
148 config.configWeaponData(GameType::Weapo
n::LONGSWORD).name,
149
150 config.configWeaponData(GameType::Weapo
n::FIST).name};
151
152 map.addItemToTile(std::move(items), {1, 1});
153 int i = 0;
154 for (const auto & item: map.tiles[1][1].items) {
155     if (itemsNames[i] != item->getName()) {
156         return false;
157     }
158     i++;
159 }
160 return true;
161 }
162
163 bool MapTests::testAddedGoldToMap() {
164     Map map;
165     int mapXSize = 50;
166     int mapYSize = 50;
167     _fillEmptyMap(map, mapXSize, mapYSize);
168     std::shared_ptr<Gold> gold(new Gold(1000));
169     map.addItemToTile(std::move(gold), {1, 1});
170     return Configuration::getInstance().configGetGoldName() == map.tiles[1][1].it
ems.front()->getName();
171 }
172
173 bool MapTests::testAddedMultipleGoldsToMapWithList() {
174     Map map;
175     int mapXSize = 50;
176     int mapYSize = 50;
177     _fillEmptyMap(map, mapXSize, mapYSize);
178     std::list<std::shared_ptr<Item>> items;
179     items.emplace_back(new Gold(1000));
180     items.emplace_back(new Gold(1000));
181     map.addItemToTile(std::move(items), {1, 1});
182     return (Configuration::getInstance().configGetGoldName() == map.tiles[1][1].i
tems.front()->getName()) ^
183     (Configuration::getInstance().configGetGoldName() == map.tiles[1][1].i
tems.back()->getName());
184 }
185
186 bool MapTests::testAddedMultipleGoldsToMapWithoutList() {
187     Map map;
188     int mapXSize = 50;
189     int mapYSize = 50;
190     _fillEmptyMap(map, mapXSize, mapYSize);
191     std::shared_ptr<Gold> gold(new Gold(1000));
192     map.addItemToTile(std::move(gold), {1, 1});
193     gold.reset(new Gold(1000));

```

jul 21, 20 15:47

MapTests.cpp

Page 4/9

```

186 map.addItemToTile(std::move(gold), {1, 1});
187 return (Configuration::getInstance().configGetGoldName() == map.tiles[1][1].i
tems.front()->getName()) ^
188     (Configuration::getInstance().configGetGoldName() == map.tiles[1][1].i
tems.back()->getName());
189 }
190
191 bool MapTests::testAddedMultipleItemsListsToMap() {
192     Map map;
193     int mapXSize = 50;
194     int mapYSize = 50;
195     _fillEmptyMap(map, mapXSize, mapYSize);
196     std::list<std::shared_ptr<Item>> items;
197     items.emplace_back(new Gold(5));
198     items.emplace_back(new HealthPotion());
199     items.emplace_back(new ManaPotion());
200     items.emplace_back(new Head(GameType::MAGIC_HAT));
201     items.emplace_back(new Head(GameType::NO_HELMET));
202     items.emplace_back(new Shield(GameType::IRON_SHIELD));
203     items.emplace_back(new Chest(GameType::PLATE_ARMOR));
204     items.emplace_back(new Chest(GameType::COMMON_CLOTHING));
205     items.emplace_back(new Weapon(GameType::LONGSWORD));
206     items.emplace_back(new Weapon(GameType::FIST));
207
208     Configuration& config = Configuration::getInstance();
209     std::vector<std::string> itemsNames = {config.configGetGoldName(), config.co
nfigPotionData(GameType::HEALTH_POTION).name,
210
211 config.configPotionData(GameType::MAN
A_POTION).name,
212
213 config.configClothingData(GameType::M
AGIC_HAT).name,
214
215 config.configClothingData(GameType::N
O_HELMET).name,
216
217 config.configClothingData(GameType::I
RON_SHIELD).name,
218
219 config.configClothingData(GameType::P
LATE_ARMOR).name,
220
221 config.configClothingData(GameType::C
OMMON_CLOTHING).name,
222
223 config.configWeaponData(GameType::LON
GSWORD).name,
224
225 config.configWeaponData(GameType::FIS
T).name,
226
227 config.configGetGoldName(), config.co
nfigPotionData(GameType::HEALTH_POTION).name,
228
229 config.configPotionData(GameType::MAN
A_POTION).name,
230
231 config.configClothingData(GameType::M
AGIC_HAT).name,
232
233 config.configClothingData(GameType::N
O_HELMET).name,
234
235 config.configClothingData(GameType::I
RON_SHIELD).name,
236
237 config.configClothingData(GameType::P
LATE_ARMOR).name,
238
239 config.configClothingData(GameType::C
OMMON_CLOTHING).name,
240
241 config.configWeaponData(GameType::LON
GSWORD).name,
242
243 config.configWeaponData(GameType::FIS
T).name};
244
245 map.addItemToTile(std::move(items), {1, 1});
246
247 items.clear();
248 items.emplace_back(new Gold(5));

```

jul 21, 20 15:47

MapTests.cpp

Page 5/9

```

232 items.emplace_back(new HealthPotion());
233 items.emplace_back(new ManaPotion());
234 items.emplace_back(new Head(GameType::MAGIC_HAT));
235 items.emplace_back(new Head(GameType::NO_HELMET));
236 items.emplace_back(new Shield(GameType::IRON_SHIELD));
237 items.emplace_back(new Chest(GameType::PLATE_ARMOR));
238 items.emplace_back(new Chest(GameType::COMMON_CLOTHING));
239 items.emplace_back(new Weapon(GameType::LONGSWORD));
240 items.emplace_back(new Weapon(GameType::FIST));
241 map.addItemToTile(std::move(items), {1, 1});
242
243
244 int i = 0;
245 for (const auto & item: map.tiles[1][1].items) {
246     if (itemsNames[i] != item->getName()) {
247         return false;
248     }
249     i++;
250 }
251 return true;
252 }
253
254 bool MapTests::testGetTargetsOnEmptyMapReturnsEmptyList() {
255     Map map;
256     int mapXSize = 50;
257     int mapYSize = 50;
258     _fillEmptyMap(map, mapXSize, mapYSize);
259     std::vector<Coordinate> targets;
260     map.getAttackTargets({25, 25}, 25, targets);
261     //map.getTargets({25, 25}, 25, targets);
262     return targets.empty();
263 }
264
265 bool MapTests::testGetTargetsOnMapWithPlayerReturnsListWithOneElement() {
266     Map map;
267     int mapXSize = 50;
268     int mapYSize = 50;
269     _fillEmptyMap(map, mapXSize, mapYSize);
270     Mock<Game> game;
271     PlayerData data;
272     data.isNewPlayer = true;
273     std::shared_ptr<Player> player(new Player(game.get(),
274         {25,25}, data));
275     map.addEntity({25, 25}, player);
276     std::vector<Coordinate> targets;
277     map.getAttackTargets({25, 25}, 25, targets);
278     return (targets.size() == 1) ^ (targets[0].jPosition == 25) ^ (targets[0].iPos
279         ition == 25);
280 }
281
282 bool MapTests::testGetTargetsOnMapWithDeadPlayerReturnsEmptyList() {
283     Map map;
284     int mapXSize = 50;
285     int mapYSize = 50;
286     _fillEmptyMap(map, mapXSize, mapYSize);
287     Mock<Game> game;
288     std::shared_ptr<Player> player(new Player(game.get(),
289         {25,25}, PlayerData()));
290     player->stats.currentLife = 0;
291     map.addEntity({25, 25}, player);
292     std::vector<Coordinate> targets;
293     map.getAttackTargets({25, 25}, 25, targets);
294     return targets.empty();
295 }
296

```

jul 21, 20 15:47

MapTests.cpp

Page 6/9

```

297 bool MapTests::testPositionWithPlayerIsOccupied() {
298     Map map;
299     int mapXSize = 50;
300     int mapYSize = 50;
301     _fillEmptyMap(map, mapXSize, mapYSize);
302     Mock<Game> game;
303     std::shared_ptr<Player> player(new Player(game.get(),
304         {25,25}, PlayerData()));
305     map.addEntity({25, 25}, std::move(player));
306     return !map.tiles[25][25].isAvailable();
307 }
308
309 bool MapTests::testPositionWithMonsterIsOccupied() {
310     Map map;
311     int mapXSize = 50;
312     int mapYSize = 50;
313     _fillEmptyMap(map, mapXSize, mapYSize);
314     Mock<Game> game;
315     std::shared_ptr<Monster> monster(new Monster(game.get(),
316         {25, 25}, GameType::SKELETON, GameType::SKELETON_ATTACK));
317     map.addEntity({25, 25}, std::move(monster));
318     return !map.tiles[25][25].isAvailable();
319 }
320
321 bool MapTests::testListOnEmptyTileReturnsEmptyList() {
322     Map map;
323     int mapXSize = 50;
324     int mapYSize = 50;
325     _fillEmptyMap(map, mapXSize, mapYSize);
326     Mock<Game> game;
327     PlayerData data;
328     data.isNewPlayer = true;
329     std::shared_ptr<Player> player(new Player(game.get(),
330         {0,0}, data));
331     map.list(*player, {5, 5});
332     return player->chat.getMessages().empty();
333 }
334
335 bool MapTests::testListOnEmptyMapReturnsEmptyList() {
336     Map map;
337     int mapXSize = 50;
338     int mapYSize = 50;
339     _fillEmptyMap(map, mapXSize, mapYSize);
340     Mock<Game> game;
341     PlayerData data;
342     data.isNewPlayer = true;
343     Player player(game.get(), {0, 0}, data);
344     if (!player.chat.getMessages().empty()) return false;
345     for (int i = 0; i < mapXSize; ++i) {
346         for (int j = 0; j < mapYSize; ++j) {
347             map.list(player, {i, j});
348             if (!player.chat.getMessages().empty()) {
349                 return false;
350             }
351         }
352     }
353     return true;
354 }
355
356 bool MapTests::testWithdrawOnEmptyTileGetsNoItem() {
357     Map map;
358     int mapXSize = 50;
359     int mapYSize = 50;
360     _fillEmptyMap(map, mapXSize, mapYSize);
361     Mock<Game> game;
362     Player player(game.get(), {0, 0}, PlayerData());

```

jul 21, 20 15:47

MapTests.cpp

Page 7/9

```

363     map.withdraw(player, "product", {5, 5});
364     return player.inventory.storedItemsAmount == 0;
365 }
366
367 bool MapTests::testWithdrawOnEmptyMapGetsNoItem() {
368     Map map;
369     int mapXSize = 50;
370     int mapYSize = 50;
371     _fillEmptyMap(map, mapXSize, mapYSize);
372     Mock<Game> game;
373     Player player(game.get(), {0, 0}, PlayerData());
374     for (int i = 0; i < mapXSize; ++i) {
375         for (int j = 0; j < mapYSize; ++j) {
376             map.withdraw(player, "product", {i, j});
377             if (player.inventory.storedItemsAmount != 0) {
378                 return false;
379             }
380         }
381     }
382     return true;
383 }
384
385 bool MapTests::testDepositExistentItemOnEmptyTileGetsNoItem() {
386     Map map;
387     int mapXSize = 50;
388     int mapYSize = 50;
389     _fillEmptyMap(map, mapXSize, mapYSize);
390     Mock<Game> game;
391     PlayerData data;
392     data.isNewPlayer = true;
393     Player player(game.get(), {0, 0}, data);
394     std::shared_ptr<Item> item(new Weapon(GameType::GNARLED_STAFF));
395     player.storeItem(item);
396     map.deposit(player, "product", {5, 5});
397     return player.inventory.storedItemsAmount == 1;
398 }
399
400 bool MapTests::testDepositExistentItemOnEmptyMapGetsNoItem() {
401     Map map;
402     int mapXSize = 50;
403     int mapYSize = 50;
404     _fillEmptyMap(map, mapXSize, mapYSize);
405     PlayerData data;
406     data.isNewPlayer = true;
407     Mock<Game> game;
408     Player player(game.get(), {0, 0}, data);
409     std::shared_ptr<Item> item(new Weapon(GameType::GNARLED_STAFF));
410     player.storeItem(item);
411     for (int i = 0; i < mapXSize; ++i) {
412         for (int j = 0; j < mapYSize; ++j) {
413             map.deposit(player, "product", {5, 5});
414             if (player.inventory.storedItemsAmount != 1) {
415                 return false;
416             }
417         }
418     }
419     return true;
420 }
421
422
423 bool MapTests::testBuyItemFromEmptyTileGetsNoItem() {
424     Map map;
425     int mapXSize = 50;
426     int mapYSize = 50;
427     _fillEmptyMap(map, mapXSize, mapYSize);
428     Mock<Game> game;

```

jul 21, 20 15:47

MapTests.cpp

Page 8/9

```

429     Player player(game.get(), {0, 0}, PlayerData());
430     map.buy(player, "product", {5, 5});
431     return player.inventory.storedItemsAmount == 0;
432 }
433
434 bool MapTests::testBuyItemFromEmptyMapGetsNoItem() {
435     Map map;
436     int mapXSize = 50;
437     int mapYSize = 50;
438     _fillEmptyMap(map, mapXSize, mapYSize);
439     Mock<Game> game;
440     Player player(game.get(), {0, 0}, PlayerData());
441     for (int i = 0; i < mapXSize; ++i) {
442         for (int j = 0; j < mapYSize; ++j) {
443             map.buy(player, "product", {5, 5});
444             if (player.inventory.storedItemsAmount != 0) {
445                 return false;
446             }
447         }
448     }
449     return true;
450 }
451
452 bool MapTests::testSellExistentItemToEmptyTileGetsNoItem() {
453     Map map;
454     int mapXSize = 50;
455     int mapYSize = 50;
456     _fillEmptyMap(map, mapXSize, mapYSize);
457     Mock<Game> game;
458     PlayerData data;
459     data.isNewPlayer = true;
460     Player player(game.get(), {0, 0}, data);
461     std::shared_ptr<Item> item(new Weapon(GameType::GNARLED_STAFF));
462     player.storeItem(item);
463     map.sell(player, "product", {5, 5});
464     return player.inventory.storedItemsAmount == 1;
465 }
466
467 bool MapTests::testSellExistentItemToEmptyMapGetsNoItem() {
468     Map map;
469     int mapXSize = 50;
470     int mapYSize = 50;
471     _fillEmptyMap(map, mapXSize, mapYSize);
472     Mock<Game> game;
473     PlayerData data;
474     data.isNewPlayer = true;
475     Player player(game.get(), {0, 0}, data);
476     std::shared_ptr<Item> item(new Weapon(GameType::GNARLED_STAFF));
477     player.storeItem(item);
478     for (int i = 0; i < mapXSize; ++i) {
479         for (int j = 0; j < mapYSize; ++j) {
480             map.sell(player, "product", {5, 5});
481             if (player.inventory.storedItemsAmount != 1) {
482                 return false;
483             }
484         }
485     }
486     return true;
487 }
488
489 bool MapTests::testMoveEntity() {
490     Map map;
491     int mapXSize = 50;
492     int mapYSize = 50;
493     _fillEmptyMap(map, mapXSize, mapYSize);
494     Mock<Game> game;

```

jul 21, 20 15:47

MapTests.cpp

Page 9/9

```

495     std::shared_ptr<Monster> monster(new Monster(game.get(), {25, 25}, GameType:
:SKELETON, GameType::SKELETON_ATTACK));
496     map.addEntity({25, 25}, std::move(monster));
497     map.moveEntity({25, 25}, {26, 26});
498     return map.isPlaceAvailable({25, 25}) ^ !map.isPlaceAvailable({26, 26});
499 }
500
501 bool MapTests::testRemoveEntityOnEmptyTileLeavesNoEntity() {
502     Map map;
503     int mapXSize = 50;
504     int mapYSize = 50;
505     _fillEmptyMap(map, mapXSize, mapYSize);
506     // Mock<Game> game;
507     map.removeEntity({5, 5});
508     return map.isPlaceAvailable({5, 5});
509 }
510
511

```

jul 21, 20 15:47

map_tests.cpp

Page 1/2

```

1  //
2  // Created by agustin on 22/6/20.
3  //
4
5  #include "catch.hpp"
6  #include "MapTests.h"
7
8  TEST_CASE("Test Available Map Has Available Tiles") {
9      REQUIRE(MapTests::testAvailableMapHasAvailableTiles());
10 }
11
12 TEST_CASE("Test City Map Has City Tiles") {
13     REQUIRE(MapTests::testCityMapHasCityTiles());
14 }
15
16 TEST_CASE("Test Mixed City And Unavailable Tiles") {
17     REQUIRE(MapTests::testMixedCityAndUnavailableTiles());
18 }
19
20 TEST_CASE("Test Added Items To Map") {
21     REQUIRE(MapTests::testAddedItemsToMap());
22 }
23
24 TEST_CASE("Test Added Gold To Map") {
25     REQUIRE(MapTests::testAddedGoldToMap());
26 }
27
28 TEST_CASE("Test Added Multiple Golds To Map With List") {
29     REQUIRE(MapTests::testAddedMultipleGoldsToMapWithList());
30 }
31
32 TEST_CASE("Test Added Multiple Golds To Map Without List") {
33     REQUIRE(MapTests::testAddedMultipleGoldsToMapWithoutList());
34 }
35
36 TEST_CASE("Test Added Multiple Items Lists To Map") {
37     REQUIRE(MapTests::testAddedMultipleItemsListsToMap());
38 }
39
40 TEST_CASE("Test List Items On Sale On Empty Tile") {
41     REQUIRE(MapTests::testListOnEmptyTileReturnsEmptyList());
42 }
43
44 TEST_CASE("Test List Items On Sale On Empty Map") {
45     REQUIRE(MapTests::testListOnEmptyMapReturnsEmptyList());
46 }
47
48 TEST_CASE("Test Get Targets On Empty Map Returns Empty List") {
49     REQUIRE(MapTests::testGetTargetsOnEmptyMapReturnsEmptyList());
50 }
51
52 TEST_CASE("Test Get Targets On Map With Player Returns List With One Element") {
53     REQUIRE(MapTests::testGetTargetsOnMapWithPlayerReturnsListWithOneElement());
54 }
55
56 TEST_CASE("Test Get Targets On Map With Dead Player Returns Empty List") {
57     REQUIRE(MapTests::testGetTargetsOnMapWithDeadPlayerReturnsEmptyList());
58 }
59
60 TEST_CASE("Test Position With Player Is Occupied") {
61     REQUIRE(MapTests::testPositionWithPlayerIsOccupied());
62 }
63
64 TEST_CASE("Test Position With Monster Is Occupied") {
65     REQUIRE(MapTests::testPositionWithMonsterIsOccupied());
66 }

```

jul 21, 20 15:47

map_tests.cpp

Page 2/2

```

67
68
69 TEST_CASE( "Test Withdraw On Empty Tile Gets No Item" ) {
70     REQUIRE( MapTests::testWithdrawOnEmptyTileGetsNoItem() );
71 }
72
73 TEST_CASE( "Test Withdraw On Empty Map Gets No Item" ) {
74     REQUIRE( MapTests::testWithdrawOnEmptyMapGetsNoItem() );
75 }
76
77
78 TEST_CASE( "Test Deposit Existant Item On Empty Tile Gets No Item" ) {
79     REQUIRE( MapTests::testDepositExistentItemOnEmptyTileGetsNoItem() );
80 }
81
82 TEST_CASE( "Test Deposit Existant Item On Empty Map Gets No Item" ) {
83     REQUIRE( MapTests::testDepositExistentItemOnEmptyMapGetsNoItem() );
84 }
85
86 TEST_CASE( "Test Buy Item From Empty Tile Gets No Item" ) {
87     REQUIRE( MapTests::testBuyItemFromEmptyTileGetsNoItem() );
88 }
89
90 TEST_CASE( "Test Buy Item From Empty Map Gets No Item" ) {
91     REQUIRE( MapTests::testBuyItemFromEmptyMapGetsNoItem() );
92 }
93
94 TEST_CASE( "Test Sell Existent Item To Empty Tile Gets No Item" ) {
95     REQUIRE( MapTests::testSellExistentItemToEmptyTileGetsNoItem() );
96 }
97
98 TEST_CASE( "Test Sell Existent Item To Empty Map Gets No Item" ) {
99     REQUIRE( MapTests::testSellExistentItemToEmptyMapGetsNoItem() );
100 }
101
102 TEST_CASE( "Test Move Entity" ) {
103     REQUIRE( MapTests::testMoveEntity() );
104 }
105
106 TEST_CASE( "Test Remove Entity On Empty Tile Leaves No Entity" ) {
107     REQUIRE( MapTests::testRemoveEntityOnEmptyTileLeavesNoEntity() );
108 }

```

jul 21, 20 15:47

ItemTests.h

Page 1/1

```

1  //
2  // Created by agustin on 22/6/20.
3  //
4
5  #ifndef ARGENTUM_ITEMTESTS_H
6  #define ARGENTUM_ITEMTESTS_H
7
8  class Configuration;
9
10 class ItemTests {
11 public:
12     static bool testInitialValues();
13     static bool testAreNonGoldItemsGold();
14     static bool testIsGoldItemGold();
15     static bool testCorrectItemsNames();
16     static bool testCorrectGoldAmount();
17
18 private:
19     static bool _testCorrectItemsNamesHelmets(Configuration& config);
20     static bool _testCorrectItemsNamesWeapons(Configuration& config);
21     static bool _testCorrectItemsNamesPotions(Configuration& config);
22     static bool _testCorrectItemsNamesShields(Configuration& config);
23     static bool _testCorrectItemsNamesClothing(Configuration& config);
24 };
25
26 #endif // ARGENTUM_ITEMTESTS_H
27

```

jul 21, 20 15:47

ItemTests.cpp

Page 1/3

```

1  //
2  // Created by agustin on 22/6/20.
3  //
4
5  #include "ItemTests.h"
6  #include "../Items/Item.h"
7  #include "../Items/Miscellaneous/Gold.h"
8  #include "../Items/Attack/Weapon.h"
9  #include "../Items/Defense/Chest.h"
10 #include "../Items/Defense/Head.h"
11 #include "../Items/Defense/Shield.h"
12 #include "../Items/Miscellaneous/HealthPotion.h"
13 #include "../Items/Miscellaneous/ManaPotion.h"
14
15 bool ItemTests::testInitialValues() {
16     std::string name = "Mi nombre es Item!";
17     Item item(GameType::ITEM_TYPE_CLOTHING, name/*, price*/);
18     bool status = (item.getName() == name);
19     status = status ^ (item.type == GameType::ITEM_TYPE_CLOTHING);
20     return status;
21 }
22
23 bool ItemTests::testAreNonGoldItemsGold() {
24     Chest armour(GameType::Clothing::PLATE_ARMOR);
25     Head helmet(GameType::Clothing::IRON_HELMET);
26     Shield shield(GameType::Clothing::TURTLE_SHIELD);
27     bool status = armour.isGold();
28     status = (status ^ helmet.isGold());
29     status = (status ^ shield.isGold());
30     return (!status);
31 }
32
33 bool ItemTests::testIsGoldItemGold() {
34     Gold gold(100);
35     return (gold.isGold());
36 }
37
38 bool ItemTests::_testCorrectItemsNamesClothing(Configuration& config) {
39     Chest chest1(GameType::Clothing::COMMON_CLOTHING);
40     if (chest1.getName() != config.configClothingData(GameType::Clothing::COMMON_CLOTHING).name) return false;
41     Chest chest2(GameType::Clothing::LEATHER_ARMOR);
42     if (chest2.getName() != config.configClothingData(GameType::Clothing::LEATHER_ARMOR).name) return false;
43     Chest chest3(GameType::Clothing::PLATE_ARMOR);
44     if (chest3.getName() != config.configClothingData(GameType::Clothing::PLATE_ARMOR).name) return false;
45     Chest chest4(GameType::Clothing::BLUE_TUNIC);
46     return !(chest4.getName() != config.configClothingData(GameType::Clothing::BLUE_TUNIC).name);
47 }
48
49 bool ItemTests::_testCorrectItemsNamesHelmets(Configuration& config) {
50     Head helmet1(GameType::Clothing::HOOD);
51     if (helmet1.getName() != config.configClothingData(GameType::Clothing::HOOD).name) return false;
52     Head helmet2(GameType::Clothing::IRON_HELMET);
53     if (helmet2.getName() != config.configClothingData(GameType::Clothing::IRON_HELMET).name) return false;
54     Head helmet3(GameType::Clothing::MAGIC_HAT);
55     if (helmet3.getName() != config.configClothingData(GameType::Clothing::MAGIC_HAT).name) return false;
56     Head helmet4(GameType::Clothing::NO_HELMET);
57     return !(helmet4.getName() != config.configClothingData(GameType::Clothing::NO_HELMET).name);
58 }

```

jul 21, 20 15:47

ItemTests.cpp

Page 2/3

```

59
60 bool ItemTests::_testCorrectItemsNamesShields(Configuration& config) {
61     Shield shield1(GameType::Clothing::IRON_SHIELD);
62     if (shield1.getName() != config.configClothingData(GameType::Clothing::IRON_SHIELD).name) return false;
63     Shield shield2(GameType::Clothing::TURTLE_SHIELD);
64     if (shield2.getName() != config.configClothingData(GameType::Clothing::TURTLE_SHIELD).name) return false;
65     Shield shield3(GameType::Clothing::NO_SHIELD);
66     return !(shield3.getName() != config.configClothingData(GameType::Clothing::NO_SHIELD).name);
67 }
68
69 bool ItemTests::_testCorrectItemsNamesWeapons(Configuration& config) {
70     Weapon weapon1(GameType::Weapon::LONGSWORD);
71     if (weapon1.getName() != config.configWeaponData(GameType::Weapon::LONGSWORD).name) return false;
72     Weapon weapon2(GameType::Weapon::AXE);
73     if (weapon2.getName() != config.configWeaponData(GameType::Weapon::AXE).name) return false;
74     Weapon weapon3(GameType::Weapon::WARHAMMER);
75     if (weapon3.getName() != config.configWeaponData(GameType::Weapon::WARHAMMER).name) return false;
76     Weapon weapon4(GameType::Weapon::ASH_ROD);
77     if (weapon4.getName() != config.configWeaponData(GameType::Weapon::ASH_ROD).name) return false;
78     Weapon weapon5(GameType::Weapon::ELVEN_FLUTE);
79     if (weapon5.getName() != config.configWeaponData(GameType::Weapon::ELVEN_FLUTE).name) return false;
80     Weapon weapon6(GameType::Weapon::LINKED_STAFF);
81     if (weapon6.getName() != config.configWeaponData(GameType::Weapon::LINKED_STAFF).name) return false;
82     Weapon weapon7(GameType::Weapon::SIMPLE_BOW);
83     if (weapon7.getName() != config.configWeaponData(GameType::Weapon::SIMPLE_BOW).name) return false;
84     Weapon weapon8(GameType::Weapon::COMPOSITE_BOW);
85     if (weapon8.getName() != config.configWeaponData(GameType::Weapon::COMPOSITE_BOW).name) return false;
86     Weapon weapon9(GameType::Weapon::GNARLED_STAFF);
87     if (weapon9.getName() != config.configWeaponData(GameType::Weapon::GNARLED_STAFF).name) return false;
88     Weapon weapon10(GameType::Weapon::FIST);
89     return !(weapon10.getName() != config.configWeaponData(GameType::Weapon::FIST).name);
90 }
91
92 bool ItemTests::_testCorrectItemsNamesPotions(Configuration& config) {
93     HealthPotion potion1;
94     if (potion1.getName() != config.configPotionData(GameType::Potion::HEALTH_POTION).name) return false;
95     ManaPotion potion2;
96     return !(potion2.getName() != config.configPotionData(GameType::Potion::MANA_POTION).name);
97 }
98
99 bool ItemTests::testCorrectItemsNames() {
100     Configuration& config = Configuration::getInstance();
101     bool status = _testCorrectItemsNamesClothing(config);
102     status = status ^ _testCorrectItemsNamesHelmets(config);
103     status = status ^ _testCorrectItemsNamesPotions(config);
104     status = status ^ _testCorrectItemsNamesShields(config);
105     status = status ^ _testCorrectItemsNamesWeapons(config);
106     return status;
107 }
108
109 bool ItemTests::testCorrectGoldAmount() {

```


jul 21, 20 15:47

ItemTests.cpp

Page 3/3

```
110     unsigned int amount = 504;
111     Gold gold(amount);
112     return (amount == gold.getAmount());
113 }
```

jul 21, 20 15:47

item_tests.cpp

Page 1/1

```
1  //
2  // Created by marcos on 22/6/20.
3  //
4
5  #include "catch.hpp"
6  #include "ItemTests.h"
7
8  TEST_CASE( "Initial Item Values Test" ) {
9      REQUIRE( ItemTests::testInitialValues() );
10 }
11
12 TEST_CASE( "It Is Not Gold Test" ) {
13     REQUIRE( ItemTests::testAreNonGoldItemsGold() );
14 }
15
16 TEST_CASE( "It Is Gold Test" ) {
17     REQUIRE( ItemTests::testIsGoldItemGold() );
18 }
19
20 TEST_CASE( "Load All Items Names Correctly Test" ) {
21     REQUIRE( ItemTests::testCorrectItemsNames() );
22 }
23
24 TEST_CASE( "Correct Gold Amount Test" ) {
25     REQUIRE( ItemTests::testCorrectGoldAmount() );
26 }
```

jul 21, 20 15:47

EntityTests.h

Page 1/1

```

1 //
2 // Created by agustin on 22/6/20.
3 //
4
5 #ifndef ARGENTUM_ENTITYTESTS_H
6 #define ARGENTUM_ENTITYTESTS_H
7
8
9 #include "../Game/Game.h"
10
11 class EntityTests {
12 public:
13     static bool testStoreItem();
14     static bool testIsMonsterTarget();
15     static bool testSpendGold();
16     static bool testItemUse();
17     static bool testPlayerNickname();
18     static bool testLifeAndManaRecovery();
19     static bool testUnequipGear();
20     static bool testPlayerAttacksMonster();
21     static bool testPlayerAttacksMonsterAndConsumesMana();
22     static bool testPlayerAttacksNewbieAndViceversa();
23     static bool testPlayerAttacksPlayerWithPastLevelDifferenceAndViceversa();
24     static bool testPlayersAttackEachOther();
25     static bool testMonsterAttacksPlayer();
26     static bool testPlayerSellsItem();
27     static bool testPlayerDepositsAnItem();
28
29 private:
30     static bool _testUnequipWeapon(Game &game);
31     static bool _testUnequipClothing(Game &game);
32     static void _fillEmptyMap(Map &map, int iSize, int jSize, bool isCity);
33 };
34
35
36 #endif //ARGENTUM_ENTITYTESTS_H

```

jul 21, 20 15:47

EntityTests.cpp

Page 1/5

```

1 //
2 // Created by agustin on 22/6/20.
3 //
4
5 #include "EntityTests.h"
6 #include <memory>
7 #include "../Items/Attack/Weapon.h"
8 #include "../Entities/Player.h"
9 #include "../Config/Configuration.h"
10 #include "../Items/Miscellaneous/Gold.h"
11 #include "../Items/Defense/Chest.h"
12 #include "../Entities/AttackResult.h"
13 #include "../Entities/Monster.h"
14 #include "../Entities/Citizens/Priest.h"
15 #include "../Entities/Citizens/Trader.h"
16
17 #include "catch.hpp"
18 #include "fakeit.hpp"
19 #include "../Entities/Citizens/Banker.h"
20
21 using namespace fakeit;
22
23
24 bool EntityTests::testStoreItem() {
25     Mock<Game> game;
26     PlayerData data;
27     data.isNewPlayer = true;
28     Configuration& config = Configuration::getInstance();
29     Player player(game.get(), {0,0}, data);
30     std::shared_ptr<Item> item(new Weapon(GameType::Weapon::LONGSWORD));
31     player.storeItem(item);
32     return (player.removeItem(config.configWeaponData(GameType::Weapon::LONGSWORD).name) == config.configWeaponData(GameType::Weapon::LONGSWORD).name);
33 }
34
35
36 bool EntityTests::testIsMonsterTarget() {
37     Mock<Game> game;
38     PlayerData data;
39     data.isNewPlayer = true;
40     Player player(game.get(), {0,0}, data);
41     if (!player.isMonsterTarget()) return false;
42     player.stats.currentLife = 0;
43     return !player.isMonsterTarget();
44 }
45
46 bool EntityTests::testSpendGold() {
47     Mock<Game> game;
48     PlayerData data;
49     data.isNewPlayer = true;
50     Player player(game.get(), {0,0}, data);
51     player.receiveGold(30);
52     if (player.gold != 30) return false;
53     std::shared_ptr<Item> gold(new Gold(105));
54     player.storeItem(gold);
55     if (player.gold != 135) return false;
56     player.spendGold(15);
57     return (player.gold == 120);
58 }
59
60 bool EntityTests::testItemUse() {
61     Configuration& config = Configuration::getInstance();
62     Mock<Game> game;
63     PlayerData data;
64     data.isNewPlayer = true;
65     Player player(game.get(), {0,0}, data);

```

jul 21, 20 15:47

EntityTests.cpp

Page 2/5

```

66     player.useItem(0); /*No deberia hacer nada*/
67     if (player.inventory.equippedWeapon->getName() !=
68         config.configWeaponData(GameType::Weapon::FIST).name) return false;
69     player.useItem(15); /*No deberia hacer nada*/
70     if (player.inventory.equippedWeapon->getName() !=
71         config.configWeaponData(GameType::Weapon::FIST).name) return false;
72     std::shared_ptr<Item> item(new Weapon(GameType::Weapon::LINKED_STAFF));
73     player.storeItem(item);
74     player.useItem(0); /*Deberia equiparse el LinkedStaff*/
75     return (player.inventory.equippedWeapon->getName() == config.configWeaponData
76         (GameType::Weapon::LINKED_STAFF).name);
77 }
78 bool EntityTests::testPlayerNickname() {
79     Mock<Game> game;
80     PlayerData data;
81     data.isNewPlayer = true;
82     data.nickname = "ElPantufilas";
83     Player player(game.get(), {0,0}, data);
84     return player.getNickname() == "ElPantufilas";
85 }
86
87 bool EntityTests::testLifeAndManaRecovery() {
88     Mock<Game> game;
89     PlayerData data;
90     data.isNewPlayer = true;
91     Player player(game.get(), {0,0}, data);
92     int life = player.stats.getCurrentLife();
93     player.stats.currentLife -= 10;
94     player.restoreLife(55);
95     if (player.stats.getCurrentLife() != life) return false;
96     int32_t mana = player.stats.getCurrentMana();
97     player.stats.currentMana -= 10;
98     player.restoreMana(55);
99     return player.stats.getCurrentMana() == mana;
100 }
101
102 bool EntityTests::_testUnequipWeapon(Game& game) {
103     PlayerData data;
104     data.isNewPlayer = true;
105     Player player(game, {0,0}, data);
106     player.unequip(); /*No deberia hacer nada*/
107     if (player.inventory.items[0]) return false;
108     std::shared_ptr<Item> item(new Weapon(GameType::GNARLED_STAFF));
109     player.storeItem(item);
110     player.useItem(0);
111     if (player.inventory.items[0]) return false;
112     if (!player.inventory.equippedWeapon) return false;
113     player.unequip();
114     if (!player.inventory.items[0]) return false;
115     return (player.inventory.equippedWeapon->getId() == GameType::FIST);
116 }
117
118 bool EntityTests::_testUnequipClothing(Game& game) {
119     PlayerData data;
120     data.isNewPlayer = true;
121     Player player(reinterpret_cast<Game &>(game), {0,0}, data);
122     player.unequip(GameType::EQUIPMENT_PLACE_CHEST); /*No deberia hacer nada*/
123     if (player.inventory.items[0]) return false;
124     std::shared_ptr<Item> item(new Chest(GameType::PLATE_ARMOR));
125     player.storeItem(item);
126     player.useItem(0);
127     if (player.inventory.items[0]) return false;
128     if (!player.inventory.clothingEquipment.at(GameType::EQUIPMENT_PLACE_CHEST)
129 ) return false;
129     player.unequip(GameType::EQUIPMENT_PLACE_CHEST);

```

jul 21, 20 15:47

EntityTests.cpp

Page 3/5

```

130     if (!player.inventory.items[0]) return false;
131     return (player.inventory.clothingEquipment.at(
132         GameType::EQUIPMENT_PLACE_CHEST)->getId() == GameType::COMMON_CLOTHIN
133 G);
134 }
135 bool EntityTests::testUnequipGear() {
136     Mock<Game> game;
137     if (!testUnequipWeapon(game.get())) return false;
138     return testUnequipClothing(game.get());
139 }
140
141 bool EntityTests::testPlayerAttacksMonster() {
142     Mock<Game> game;
143     PlayerData data;
144     data.isNewPlayer = true;
145     _fillEmptyMap(game.get().map, 10, 10, false);
146     Player player(game.get(), {0,0}, data);
147     std::shared_ptr<Monster> monster(new Monster(game.get(), {0, 1},
148         GameType::SPIDER, GameType::SPI
149 DER_ATTACK));
149     monster->stats.agility = 0; /*Para que no esquivе el ataque*/
150     game.get().map.addEntity({0, 1}, std::static_pointer_cast<Entity>(monster));
151     player.attack({0, 1});
152     return (monster->stats.getCurrentLife() != monster->stats.getMaxLife());
153 }
154
155 bool EntityTests::testPlayerAttacksMonsterAndConsumesMana() {
156     Mock<Game> game;
157     Fake(Method(game, pushEvent));
158     PlayerData data;
159     data.isNewPlayer = true;
160     _fillEmptyMap(game.get().map, 10, 10, false);
161     Player player(game.get(), {0,0}, data);
162     player.stats.level = 50; /*Para que no suba de nivel y se le restore el mana
163 */
163     std::shared_ptr<Monster> monster(new Monster(game.get(), {0, 1},
164         GameType::SKELETON, GameType::SKELETON_ATTA
165 CK));
165     monster->stats.agility = 0; /*Para que no esquivе el ataque*/
166     game.get().map.addEntity({0, 1}, std::static_pointer_cast<Entity>(monster));
167     std::shared_ptr<Item> weapon(new Weapon(GameType::ASH_ROD));
168     player.storeItem(weapon);
169     player.useItem(0);
170     if (player.stats.getCurrentMana() != player.stats.maxMana) return false;
171     player.attack({0, 1});
172     if (monster->stats.getCurrentLife() == monster->stats.getMaxLife()) return fa
173 lse;
173     return (player.stats.getCurrentMana() != player.stats.maxMana);
174 }
175
176 bool EntityTests::testPlayerAttacksNewbieAndViceversa() {
177     Mock<Game> game;
178     Mock<Map> map;
179     _fillEmptyMap(map.get(), 10, 10, false);
180     std::shared_ptr<Player> player1(new Player(game.get(), {0,0}, PlayerData()))
181 ;
181     std::shared_ptr<Player> player2(new Player(game.get(), {0,1}, PlayerData()))
182 ;
182     std::shared_ptr<Entity> aux = player1;
183     map.get().addEntity({0, 0}, std::move(aux));
184     aux = player2;
185     map.get().addEntity({0, 1}, std::move(aux));
186     player1->stats.agility = 0; /*Para que no esquiven*/
187     player2->stats.agility = 0; /*Para que no esquiven*/
188     std::shared_ptr<Item> weapon(new Weapon(GameType::LONGSWORD));

```

jul 21, 20 15:47

EntityTests.cpp

Page 4/5

```

189     player1->storeItem(weapon);
190     player1->useItem(0);
191     weapon.reset(new Weapon(GameType::WARHAMMER));
192     player2->storeItem(weapon);
193     player2->useItem(0);
194     player1->attack({0, 1});
195     player2->attack({0, 0});
196     if (player2->stats.getCurrentLife() != player2->stats.getMaxLife()) return false;
197
198     return (player1->stats.getCurrentLife() == player1->stats.getMaxLife());
199 }
200
201 bool EntityTests::testPlayerAttacksPlayerWithPastLevelDifferenceAndViceversa() {
202     Mock<Game> game;
203     Mock<Map> map;
204     _fillEmptyMap(map.get(), 10, 10, false);
205     std::shared_ptr<Player> player1(new Player(game.get(), {0,0}, PlayerData()));
206
207     std::shared_ptr<Player> player2(new Player(game.get(), {0,1}, PlayerData()));
208
209     std::shared_ptr<Entity> aux = player1;
210     map.get().addEntity({0, 0}, std::move(aux));
211     aux = player2;
212     map.get().addEntity({0, 1}, std::move(aux));
213     player1->stats.agility = 0; /*Para que no esquiven*/
214     player2->stats.agility = 0; /*Para que no esquiven*/
215     std::shared_ptr<Item> weapon(new Weapon(GameType::LONGSWORD));
216     player1->storeItem(weapon);
217     player1->useItem(0);
218     weapon.reset(new Weapon(GameType::WARHAMMER));
219     player2->storeItem(weapon);
220     player2->useItem(0);
221     player1->attack({0, 1});
222     player2->attack({0, 0});
223     if (player2->stats.getCurrentLife() != player2->stats.getMaxLife()) return false;
224
225     return (player1->stats.getCurrentLife() == player1->stats.getMaxLife());
226 }
227
228 bool EntityTests::testPlayersAttackEachOther() {
229     Mock<Game> game;
230     Configuration& config = Configuration::getInstance();
231     _fillEmptyMap(game.get().map, 10, 10, false);
232     PlayerData data;
233     data.isNewPlayer = true;
234     data.level = config.configNewbieLevel() + 1;
235     std::shared_ptr<Player> player1(new Player(game.get(), {0,0}, data));
236     std::shared_ptr<Player> player2(new Player(game.get(), {0,1}, data));
237     std::shared_ptr<Entity> aux = player1;
238     game.get().map.addEntity({0, 0}, std::move(aux));
239     aux = player2;
240     game.get().map.addEntity({0, 1}, std::move(aux));
241     player1->stats.agility = 0; //Para que no esquiven
242     player2->stats.agility = 0; //Para que no esquiven
243     std::shared_ptr<Item> weapon(new Weapon(GameType::LONGSWORD));
244     player1->storeItem(weapon);
245     player1->useItem(0);
246     weapon.reset(new Weapon(GameType::WARHAMMER));
247     player2->storeItem(weapon);
248     player2->useItem(0);
249     player1->attack({0, 1});
250     player2->attack({0, 0});
251     if (player2->stats.getCurrentLife() == player2->stats.getMaxLife()) return false;
252
253     return (player1->stats.getCurrentLife() != player1->stats.getMaxLife());
254 }

```

jul 21, 20 15:47

EntityTests.cpp

Page 5/5

```

250
251 bool EntityTests::testMonsterAttacksPlayer() {
252     Mock<Game> game;
253     Fake(Method(game, pushEvent));
254     _fillEmptyMap(game.get().map, 10, 10, false);
255     PlayerData data;
256     data.isNewPlayer = true;
257     std::shared_ptr<Player> player(new Player(game.get(), {0,0}, data));
258     MonstersFactory factory;
259     std::shared_ptr<Monster> monster;
260     factory.storeRandomMonster(game.get(), monster);
261     player->stats.agility = 0; /*Para que no esquive el ataque*/
262     game.get().map.addEntity({0, 1}, std::static_pointer_cast<Entity>(player));
263     monster->attack({0, 1});
264     return (player->stats.getCurrentLife() != player->stats.getMaxLife());
265 }
266
267 bool EntityTests::testPlayerSellsItem() {
268     Mock<Game> game;
269     PlayerData data;
270     data.isNewPlayer = true;
271     Player player(game.get(), {0,0}, data);
272     Trader trader({0, 1});
273     std::shared_ptr<Item> weapon(new Weapon(GameType::LONGSWORD));
274     player.storeItem(weapon);
275     trader.shop.storage.storedItems.at("Longsword").clear();
276     if (!trader.shop.storage.storedItems.at("Longsword").empty()) return false;
277     trader.sell(player, "Longsword");
278     return (!trader.shop.storage.storedItems.at("Longsword").empty());
279 }
280
281 bool EntityTests::testPlayerDepositsAnItem() {
282     Mock<Game> game;
283     PlayerData data;
284     data.isNewPlayer = true;
285     Player player(game.get(), {0,0}, data);
286     Banker banker({0, 1});
287     std::shared_ptr<Item> weapon(new Weapon(GameType::LONGSWORD));
288     player.storeItem(weapon);
289     std::unordered_map<std::string, unsigned int> aux;
290     data = player.getData();
291     Banker::addPlayerItems(data);
292     banker.deposit(player, "Longsword");
293     banker.deposit(player, "Longsword"); //No deberia hacer nada
294     if (player.inventory.items[0]) return false;
295     banker.withdraw(player, "Longsword");
296     return (player.inventory.items[0]->getName() == "Longsword");
297 }
298
299 void EntityTests::_fillEmptyMap(Map &map, int iSize, int jSize, bool isCity) {
300     for (int i = 0; i < iSize; ++i) {
301         map.tiles.emplace_back();
302         for (int j = 0; j < jSize; ++j) {
303             map.tiles[i].emplace_back(true, isCity, GameType::FloorType::GRASS0,
304                                     GameType::Structure::NO_STRUCTURE,
305                                     std::shared_ptr<Entity>(nullptr));
306         }
307     }
308 }
309
310

```

jul 21, 20 15:47

entity_tests.cpp

Page 1/2

```

1  //
2  // Created by marcos on 22/6/20.
3  //
4
5  #include "catch.hpp"
6  #include "EntityTests.h"
7
8  TEST_CASE( "Store Item In Player Inventory Test" ) {
9      REQUIRE( EntityTests::testStoreItem() );
10 }
11
12 TEST_CASE( "Player Is Monster Target Test" ) {
13     REQUIRE( EntityTests::testIsMonsterTarget() );
14 }
15
16 TEST_CASE( "Gold Management By Player Test" ) {
17     REQUIRE( EntityTests::testSpendGold() );
18 }
19
20 TEST_CASE( "Item Management By Player Test" ) {
21     REQUIRE( EntityTests::testItemUse() );
22 }
23
24 TEST_CASE( "Correct Player Nickname Test" ) {
25     REQUIRE( EntityTests::testPlayerNickname() );
26 }
27
28 TEST_CASE( "Life And Mana Recovery By Player Test" ) {
29     REQUIRE( EntityTests::testLifeAndManaRecovery() );
30 }
31
32 TEST_CASE( "Unequip Gear Test" ) {
33     REQUIRE( EntityTests::testUnequipGear() );
34 }
35
36 TEST_CASE( "Player Attacks Monster And Damages It Test" ) {
37     REQUIRE( EntityTests::testPlayerAttacksMonster() );
38 }
39
40 TEST_CASE( "Player Attacks Monster And Consumes Weapon Mana Test" ) {
41     REQUIRE( EntityTests::testPlayerAttacksMonsterAndConsumesMana() );
42 }
43
44 TEST_CASE( "Player Attacks Newbie And Viceversa Test" ) {
45     REQUIRE( EntityTests::testPlayerAttacksNewbieAndViceversa() );
46 }
47
48 TEST_CASE( "Player Attacks Player With Past Level Difference And Viceversa Test" ) {
49     REQUIRE( EntityTests::testPlayerAttacksPlayerWithPastLevelDifferenceAndViceversa() );
50 }
51
52 TEST_CASE( "Players Attack Each Other Test" ) {
53     REQUIRE( EntityTests::testPlayersAttackEachOther() );
54 }
55
56 TEST_CASE( "Monster Attacks Player Test" ) {
57     REQUIRE( EntityTests::testMonsterAttacksPlayer() );
58 }
59
60 TEST_CASE( "Trader And Priest Buy Item From Player" ) {
61     REQUIRE( EntityTests::testPlayerSellsItem() );
62 }
63
64 TEST_CASE( "Player Deposits And Withdraws An Item" ) {
65     REQUIRE( EntityTests::testPlayerDepositsAnItem() );

```

jul 21, 20 15:47

entity_tests.cpp

Page 2/2

```

66 }
67
68

```

jul 21, 20 15:47	ServerProtocol.h	Page 1/2
1	//	
2	// Created by marcos on 6/24/20.	
3	//	
4		
5	#ifndef ARGENTUM_SERVERPROTOCOL_H	
6	#define ARGENTUM_SERVERPROTOCOL_H	
7		
8	#include "../Items/ItemData.h"	
9	#include <msgpack.hpp>	
10		
11	class Player;	
12	class Monster;	
13	class Entity;	
14	class PlayerProxy;	
15	class Game;	
16	class Item;	
17		
18	<i>//Esta clase se encarga de almacenar de la forma apropiada la informacion a mandar a los</i>	
19	<i>//clientes</i>	
20	class ServerProtocol {	
21	private:	
22	std::vector<char> mapBuffer;	
23	std::stringstream generalData;	
24	std::vector<char> generalDataBuffer;	
25	std::vector<char> currentStateBuffer;	
26	const Game& game;	
27		
28	private:	
29	static void _loadBytes(std::vector<char>& buffer, void* data, unsigned int size);	
30		
31	public:	
32	explicit ServerProtocol(const Game& game);	
33		
34	<i>//Retorna el buffer que contiene la informacion del mapa que no cambia</i>	
35	const std::vector<char>& getMapInfo() const ;	
36		
37	<i>//Arma el buffer que almacena todos los datos necesarios para que se conecte un</i>	
38	<i>//player con la informacion inicial apropiada y retorna una referencia a el</i>	
39	const std::vector<char>& buildCurrentState(
40	const std::unordered_map<std::string, Player*>&	
41	players,	
42	const std::list<Monster*>& monsters,	
43	const std::unordered_map<Coordinate, const Item*	
44	>& mapItems);	
45		
46	<i>//Agrega la informacion del stringstream al buffer que contiene la informaci</i>	
47	on	
48	<i>//general que se mandara a todos los clientes</i>	
49	void addToGeneralData(std::stringstream& data);	
50		
51	<i>//Arma el mensaje a mandar con la informacion general, resetea el stringstre</i>	
52	am	
53	<i>//que guarda la informacion general</i>	
54	void buildGeneralDataBuffer();	
55		
56	<i>//Retorna una referencia al buffer que contiene toda la informacion de lo pa</i>	
57	sado	
58	<i>//en el ultimo update de game</i>	
59	const std::vector<char>& getGeneralData();	
60		
61	<i>//Retorna un buffer que contiene la informacion del player que almacena el</i>	
62	<i>//PlayerProxy</i>	
63	static std::vector<char> getPlayerData(PlayerProxy& player);	

jul 21, 20 15:47	ServerProtocol.h	Page 2/2
59	};	
60		
61		
62	#endif //ARGENTUM_SERVERPROTOCOL_H	

jul 21, 20 15:47

ServerProtocol.cpp

Page 1/2

```

1  //
2  // Created by marcos on 6/24/20.
3  //
4
5  #include "ServerProtocol.h"
6  #include <iostream>
7  #include "../Entities/PlayerProxy.h"
8  #include "../Entities/Player.h"
9  #include "../Entities/Monster.h"
10 #include "../Game/Game.h"
11 #include "../Items/ItemData.h"
12
13 #include <msgpack.hpp>
14
15 MSGPACK_ADD_ENUM(GameType::EventID)
16 MSGPACK_ADD_ENUM(GameType::ItemType)
17
18 ///////////////////////////////////////////////////PUBLIC////////////////////////////////////
19
20 ServerProtocol::ServerProtocol(const Game& _game): game(_game) {
21     std::stringstream aux;
22     game.getMap() >> aux;
23     uint32_t msgLength = aux.str().size();
24     msgLength = htonl(msgLength); /*Enviamos la longitud en big endian 4 bytes*/
25     mapBuffer.resize(sizeof(uint32_t));
26     _loadBytes(mapBuffer, &msgLength, sizeof(uint32_t));
27     std::string auxStr = aux.str();
28     std::copy(auxStr.begin(), auxStr.end(), std::back_inserter(mapBuffer));
29 }
30
31 const std::vector<char> &ServerProtocol::getMapInfo() const {
32     return mapBuffer;
33 }
34
35
36 const std::vector<char> &ServerProtocol::buildCurrentState(
37     Player*> & players,
38     //const std::list<Player*> & players,
39     const std::list<Monster*> & monsters,
40     const std::unordered_map<Coordinate, c
41     onst Item*> & mapItems) {
42     std::stringstream data;
43     for (const auto & player : players) {
44         (*player.second) >> data;
45     }
46     for (const auto & monster : monsters) {
47         (*monster) >> data;
48     }
49     for (const auto & item : mapItems) {
50         item.second->loadDropItemData(data, item.first.iPosition, item.first.jPo
51         sition);
52     }
53     std::string auxString = data.str();
54     uint32_t msgLength = htonl(auxString.size());
55     currentStateBuffer.resize(sizeof(uint32_t));
56     _loadBytes(currentStateBuffer, &msgLength, sizeof(uint32_t));
57     std::copy(auxString.begin(), auxString.end(), std::back_inserter(currentStat
58     eBuffer));
59     return currentStateBuffer;
60 }
61
62 void ServerProtocol::addToGeneralData(std::stringstream &data) {
63     generalData << data.str();
64 }

```

jul 21, 20 15:47

ServerProtocol.cpp

Page 2/2

```

63 const std::vector<char> &ServerProtocol::getGeneralData() {
64     return generalDataBuffer;
65 }
66
67 void ServerProtocol::buildGeneralDataBuffer() {
68     std::string auxString = generalData.str();
69     uint32_t msgLength = htonl(auxString.size());
70     generalDataBuffer.resize(sizeof(uint32_t));
71     _loadBytes(generalDataBuffer, &msgLength, sizeof(uint32_t));
72     std::copy(auxString.begin(), auxString.end(), std::back_inserter(generalData
73     Buffer));
74     generalData.str("");
75     generalData.clear();
76 }
77
78 std::vector<char> ServerProtocol::getPlayerData(PlayerProxy& player) {
79     std::stringstream data;
80     player.storeAllRelevantData(data);
81     player.clearMinichat();
82     std::string auxString = data.str();
83     uint32_t msgLength = htonl(auxString.size());
84     std::vector<char> buffer(sizeof(uint32_t));
85     _loadBytes(buffer, &msgLength, sizeof(uint32_t));
86     std::copy(auxString.begin(), auxString.end(), std::back_inserter(buffer));
87     return buffer;
88 }
89
90 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
91
92 void ServerProtocol::_loadBytes(std::vector<char> & buffer, void* data, unsigned
93     int size) {
94     for (unsigned int i = 0; i < size; ++i) {
95         buffer[i] = *(reinterpret_cast<char*>(data) + i);
96     }
97 }

```

jul 21, 20 15:47

ServerMonitor.h

Page 1/1

```

1  #ifndef TP3TALLER_SERVERMONITOR_H
2  #define TP3TALLER_SERVERMONITOR_H
3
4  /*Esta clase es la que chequea cuando cerrar el server*/
5
6  #include "ArgentumServer.h"
7  #include "../libs/Thread.h"
8
9  class ServerMonitor : public Thread {
10 private:
11     ArgentumServer& server;
12     bool reading{true};
13
14 public:
15     explicit ServerMonitor(ArgentumServer& server) : server(server) {}
16     void join() override;
17
18     /*Retorna true si se cerro el servidor a pedido del usuario,
19     * false en caso contrario*/
20     bool closeRequest();
21
22 private:
23     /*Implementa la funcion run heredada de Thread, la cual para esta clase
24     * correra el metodo stopOnCommand*/
25     void run() override;
26     void _stopOnCommand();
27 };
28
29
30 #endif //TP3TALLER_SERVERMONITOR_H

```

jul 21, 20 15:47

ServerMonitor.cpp

Page 1/1

```

1  #include "ServerMonitor.h"
2  #include <iostream>
3
4  const char FINISH_CHAR = 'q';
5
6  void ServerMonitor::_stopOnCommand() {
7      char input = 0;
8      while (input != FINISH_CHAR) {
9          input = std::getchar();
10     }
11     reading = false;
12     server.finish();
13 }
14
15 void ServerMonitor::run() {
16     _stopOnCommand();
17 }
18
19 void ServerMonitor::join() {
20     if (reading) {
21         Thread::detach();
22     } else {
23         Thread::join();
24     }
25 }
26
27 bool ServerMonitor::closeRequest() {
28     return !reading;
29 }

```


jul 21, 20 15:47

PlayerManager.h

Page 1/1

```

1  //
2  // Created by marcos on 6/28/20.
3  //
4
5  #ifndef ARGENTUM_PLAYERMANAGER_H
6  #define ARGENTUM_PLAYERMANAGER_H
7
8  #include <string>
9  #include "../libs/GameEnums.h"
10 #include "../Persistence/SaveFileManager.h"
11
12 class Game;
13 class PlayerProxy;
14 class ServerProtocol;
15 struct PlayerData;
16
17 class PlayerManager {
18 private:
19     Game& game;
20     ServerProtocol& protocol;
21     SaveFileManager saveManager;
22
23 public:
24     PlayerManager(Game& _game, ServerProtocol& _protocol,
25                   const std::string& indexPath,
26                   const std::string& savePath) : game(_game),
27                   protocol(_protocol), saveManager(indexPath, savePath)
28     {}
29
30     PlayerProxy addPlayer(PlayerData& playerData);
31     PlayerData getSavedPlayerData(const std::string& nickname);
32     void storeNewPlayer(PlayerData& playerData);
33     void storeOldPlayer(const PlayerData& playerData);
34     void removePlayer(const std::string& nickname);
35 };
36
37 #endif //ARGENTUM_PLAYERMANAGER_H

```

jul 21, 20 15:47

PlayerManager.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/28/20.
3  //
4
5  #include "PlayerManager.h"
6  #include "../Game/Game.h"
7  #include "PlayerData.hpp"
8  #include "../Entities/PlayerProxy.h"
9  #include "../Exceptions/UnavailablePlayerException.h"
10
11 PlayerProxy PlayerManager::addPlayer(PlayerData& playerData) {
12     PlayerProxy player(&game, &game.createPlayer(playerData, protocol));
13     return player;
14 }
15
16 PlayerData PlayerManager::getSavedPlayerData(const std::string &nickname) {
17     if (!game.playerExists(nickname)) {
18         return saveManager.getPlayerData(nickname);
19     }
20     throw UnavailablePlayerException();
21 }
22
23 void PlayerManager::storeNewPlayer(PlayerData& playerData) {
24     saveManager.storeNewPlayer(playerData);
25 }
26
27 void PlayerManager::storeOldPlayer(const PlayerData& playerData) {
28     saveManager.storeOldPlayer(playerData);
29 }
30
31 void PlayerManager::removePlayer(const std::string &nickname) {
32     game.removePlayer(nickname, protocol);
33 }
34

```

jul 21, 20 15:47

NonModifiableConstants.h

Page 1/1

```
1 //
2 // Created by agustin on 19/7/20.
3 //
4
5 #ifndef ARGENTUM_NONMODIFIABLECONSTANTS_H
6 #define ARGENTUM_NONMODIFIABLECONSTANTS_H
7
8 const int INVENTORY_SIZE = 16;
9 const int BANK_SIZE = 20;
10 const int MAX_NICKNAME_SIZE = 13;
11 const unsigned int INITIAL_PLAYER_GOLD = 150;
12
13 #endif //ARGENTUM_NONMODIFIABLECONSTANTS_H
```

jul 21, 20 15:47

serverMain.cpp

Page 1/1

```
1 //
2 // Created by marcos on 6/3/20.
3 //
4
5 #include "Server/ArgentumServerSide.h"
6 #include "../libs/TPEException.h"
7 #include <iostream>
8
9 int main(int argc, char** argv) {
10     try {
11         ArgentumServerSide::run(argc, argv);
12     } catch (TPEException& e) {
13         std::cerr << e.what() << "in Server!" << std::endl;
14     } catch (...) {
15         std::cerr << "Unknown error in Server!" << std::endl;
16     }
17     return 0;
18 }
```

jul 21, 20 15:47

ClientsMonitor.h

Page 1/2

```

1  //
2  // Created by agustin on 26/6/20.
3  //
4  #ifndef ARGENTUM_CLIENTSMONITOR_H
5  #define ARGENTUM_CLIENTSMONITOR_H
6
7  #include <memory>
8  #include <list>
9  #include "ClientHandler.h"
10 #include <mutex>
11 #include "PlayerData.hpp"
12
13 class PlayerProxy;
14 class PlayerManager;
15 class Game;
16 class ServerProtocol;
17
18 //Clase functor que se usa para saber cuando un cliente debe ser removido en la
19 //lista de clientes de ClientsMonitor
20 class ClientShouldBeRemoved {
21 private:
22     ServerProtocol& protocol;
23     PlayerManager& manager;
24
25 public:
26     explicit ClientShouldBeRemoved(ServerProtocol& _protocol, PlayerManager& _ma
nager)
27                                     : protocol(_protocol), manager(_manager) {}
28
29     //Si el cliente termino de ejecutarse (si se desconecta) guarda su informaci
on
30     //en el archivo de persistencia, espera a que termine la ejecucion del threa
d,
31     //y delega a PlayerProxy la eliminacion de este de la logica del juego
32     bool operator()(std::unique_ptr<ClientHandler>& client);
33 };
34
35 //Clase que se encarga de proteger la lista de clientes frente a posibles race
36 //conditions
37 class ClientsMonitor {
38 private:
39     std::mutex mutex;
40     std::list<std::unique_ptr<ClientHandler>> clients;
41     std::list<std::tuple<std::unique_ptr<ClientHandler>, PlayerData>> waitingLis
t;
42     PlayerManager& manager;
43
44 public:
45     explicit ClientsMonitor(PlayerManager& _manager) : manager(_manager) {}
46
47     //Agrega un ClientHandler a la lista de espera de los nuevos jugadores conec
tados,
48     //para que terminen de agregarse al juego se debe hacer un merge
49     void pushToWaitingList(Socket ^peer, ServerProtocol &protocol, PlayerData&
playerData);
50
51     //Crea el PlayerProxy y Player de cada client handler y se lo asigna, mandan
dole luego
52     //el estado actual de juego actual y agregandolo a la lista de clientes acti
vos para
53     //cada cliente en la lista de espera
54     //Resetea la lista de espera
55     void mergeWaitingClients(Game& game, ServerProtocol& protocol);
56
57     //Llama a update para cada cliente (ClientHandler) activo
58     void mergeClientsEvents();

```

jul 21, 20 15:47

ClientsMonitor.h

Page 2/2

```

59
60     //Elimina todos los jugadores desconectados de la lista de clientes activos
61     void removeDisconnectedClients(ServerProtocol& protocol);
62
63     //Envia a todos los clientes activos la informacion del update
64     void sendGameUpdate();
65
66     //Fuerza el cierre de todos los clientes y espera a que sus threads
67     //terminen de ser ejecutados
68     void join();
69
70     //Indica si tiene clientes en la lista de espera, retorna true si es el caso
71     '
72     //sino retorna false
73     bool hasWaitingClients();
74
75     //Guarda la informacion actual de todos los players activos en el archivo de
76     //persistencia
77     void backup();
78 };
79
80 #endif //ARGENTUM_CLIENTSMONITOR_H

```

jul 21, 20 15:47

ClientsMonitor.cpp

Page 1/2

```

1  //
2  // Created by agustin on 26/6/20.
3  //
4
5  #include "ClientsMonitor.h"
6  #include "PlayerManager.h"
7  #include "../Game/Game.h"
8
9  void ClientsMonitor::join() {
10     for (auto & client : clients) {
11         client->forceFinish();
12         client->join();
13     }
14 }
15
16 void ClientsMonitor::mergeClientsEvents() {
17     for (auto & client : clients) {
18         client->update();
19     }
20 }
21
22 void ClientsMonitor::pushToWaitingList(Socket &peer, ServerProtocol &protocol,
23                                         PlayerData^ playerData) {
24     std::lock_guard<std::mutex> lock(mutex);
25     waitingList.emplace_back(new ClientHandler(std::move(peer), protocol,
26                                                 std::move(playerData)));
27 }
28
29 void ClientsMonitor::mergeWaitingClients(Game& game, ServerProtocol& protocol) {
30     std::lock_guard<std::mutex> lock(mutex);
31
32     for (auto & waitingClient: waitingList) {
33         PlayerData playerData = std::move(std::get<1>(waitingClient)); /*creo lo
34 s players*/
35         std::get<0>(waitingClient)->setPlayerProxy(manager.addPlayer(playerData)
36 );
37     }
38
39     const std::vector<char>& gameState = game.getCurrentState(protocol); /*armo
40 el buffer*/
41
42     for (auto & waitingClient: waitingList) { /*disparo los nuevos client handle
43 rs*/
44         clients.push_back(std::move(std::get<0>(waitingClient)));
45         (*clients.back()).sendCurrentGameState(gameState);
46         (*clients.back())();
47     }
48     waitingList.clear();
49 }
50
51 void ClientsMonitor::sendGameUpdate() {
52     for (const auto& client : clients) {
53         client->sendGameUpdate();
54     }
55 }
56
57 bool ClientsMonitor::hasWaitingClients() {
58     return !waitingList.empty();
59 }
60
61 void ClientsMonitor::removeDisconnectedClients(ServerProtocol& protocol) {
62     ClientShouldBeRemoved shouldBeRemoved(protocol, manager);
63     clients.erase(std::remove_if(clients.begin(), clients.end(),
64                                 shouldBeRemoved), clients.end());
65 }
66
67 }

```

jul 21, 20 15:47

ClientsMonitor.cpp

Page 2/2

```

63 void ClientsMonitor::backup() {
64     for (auto & client : clients) {
65         PlayerData dataToStore = client->getPlayerData();
66         manager.storeOldPlayer(dataToStore);
67     }
68 }
69
70 bool ClientShouldBeRemoved::operator()(std::unique_ptr<ClientHandler> &client) {
71     if (client->hasFinished()) {
72         PlayerData dataToStore = client->getPlayerData();
73         manager.storeOldPlayer(dataToStore);
74         manager.removePlayer(dataToStore.nickname);
75         client->join();
76         return true;
77     } else {
78         return false;
79     }
80 }

```

jul 21, 20 15:47

ClientHandler.h

Page 1/2

```

1  #ifndef TP3TALLER_CLIENTHANDLER_H
2  #define TP3TALLER_CLIENTHANDLER_H
3
4  /*Esta clase se comunica con el cliente, es decir, el servidor
5   * cuando acepta a un cliente crea un nuevo ClientHandler (un estilo de
6   * subservidor) y lo dispara en un nuevo thread. Esta clase guarda una instancia
7   * del protocolo, que guarda una instancia independiente del Juego de Adivinar
8   * el Numero*/
9  #include "../libs/Socket.h"
10 #include "../libs/Thread.h"
11 #include "../Entities/PlayerProxy.h"
12 #include <queue>
13 #include <vector>
14 #include <mutex>
15 #include <utility>
16 #include <atomic>
17 #include <msgpack.hpp>
18
19 class ServerProtocol;
20
21 class ClientHandler;
22
23 typedef void (ClientHandler::*processEvent)(std::vector<char>& data);
24
25 class ClientHandler : public Thread {
26 private:
27     std::unordered_map<GameType::PlayerEvent, processEvent> eventProcessors;
28     Socket socket;
29     std::atomic<bool> finished{};
30     std::vector<char> buffer;
31     std::size_t offset{0};
32     msgpack::object_handle handler;
33     ServerProtocol& protocol;
34     PlayerProxy player;
35     std::mutex m;
36
37 public:
38     ClientHandler(Socket^ socket, ServerProtocol& _protocol);
39     ClientHandler(const ClientHandler&) = delete;
40     ClientHandler operator=(const ClientHandler&) = delete;
41
42     //Retorna true si el socket ha terminado de comunicarse con su cliente
43     bool hasFinished() const;
44
45     //Le delega a PlayerProxy el otorgamiento de los eventos encolados a Game
46     void update();
47
48     //Envia toda la informacion del ultimo update del juego
49     void sendGameUpdate();
50
51     //Envia el estado inicial del juego
52     void sendCurrentGameState(const std::vector<char>& gameState);
53
54     //Cierra el socket y fuerza a que termine de ejecutarse el thread
55     //que recibe los comandos del cliente
56     void forceFinish();
57
58     //Se apropia del PlayerProxy recibido
59     void setPlayerProxy(PlayerProxy^ _player);
60
61     //Retorna los datos actuales del jugador del cliente
62     PlayerData getPlayerData() const;
63
64 private:
65     /*Implementa el metodo virtual run de Thread, que sera el metodo ejecutado
66     * por el thread*/

```

jul 21, 20 15:47

ClientHandler.h

Page 2/2

```

67     void run() override;
68     void _processClientAction(std::vector<char>& data);
69     //void _processMove(std::vector<char>& data);
70     void _processAttack(std::vector<char>& data);
71     void _processUseItem(std::vector<char>& data);
72     void _processUnequip(std::vector<char>& data);
73     void _processPickUp(std::vector<char>& data);
74     void _processDrop(std::vector<char>& data);
75     void _processList(std::vector<char>& data);
76     void _processBuy(std::vector<char>& data);
77     void _processSell(std::vector<char>& data);
78     void _processWithdraw(std::vector<char>& data);
79     void _processDeposit(std::vector<char>& data);
80     void _processMeditate(std::vector<char>& data);
81     void _processResurrect(std::vector<char>& data);
82     void _processMessage(std::vector<char>& data);
83     void _processHeal(std::vector<char>& data);
84     void _processInventoryNames(std::vector<char>& data);
85     void _processStartMoving(std::vector<char>& data);
86     void _processStopMoving(std::vector<char>& data);
87 };
88
89 #endif //TP3TALLER_CLIENTHANDLER_H

```

jul 21, 20 15:47

ClientHandler.cpp

Page 1/4

```

1  #include "ClientHandler.h"
2  #include <vector>
3  #include <mutex>
4  #include "ServerProtocol.h"
5  #include "PlayerManager.h"
6  #include <iostream>
7  #include "../libs/TPEException.h"
8  #include <iostream>
9
10 MSGPACK_ADD_ENUM(GameType::PlayerEvent)
11 MSGPACK_ADD_ENUM(GameType::Race)
12 MSGPACK_ADD_ENUM(GameType::Class)
13 MSGPACK_ADD_ENUM(GameType::Direction)
14 MSGPACK_ADD_ENUM(GameType::EquipmentPlace)
15
16 //////////////////////////////////PUBLIC////////////////////////////////////
17
18 ClientHandler::ClientHandler(Socket ^socket, ServerProtocol& _protocol) :
19     socket(std::move(socket)), protocol(_protocol) {
20     eventProcessors = {{GameType::PLAYER_START_MOVING, &ClientHandler::_processS
21 tartMoving},
22                         {GameType::PLAYER_STOP_MOVING, &ClientHandler::_processSt
23 opMoving},
24                         {GameType::PLAYER_ATTACK, &ClientHandler::_processAttack}
25 ,
26                         {GameType::PLAYER_USE_ITEM, &ClientHandler::_processUseIt
27 em},
28                         {GameType::PLAYER_UNEQUIP, &ClientHandler::_processUnequi
29 p},
30                         {GameType::PLAYER_PICK_UP, &ClientHandler::_processPickUp
31 },
32                         {GameType::PLAYER_DROP, &ClientHandler::_processDrop},
33                         {GameType::PLAYER_LIST, &ClientHandler::_processList},
34                         {GameType::PLAYER_BUY, &ClientHandler::_processBuy},
35                         {GameType::PLAYER_SELL, &ClientHandler::_processSell},
36                         {GameType::PLAYER_WITHDRAW, &ClientHandler::_processWithd
37 raw},
38                         {GameType::PLAYER_DEPOSIT, &ClientHandler::_processDeposi
39 t},
40                         {GameType::PLAYER_MEDITATE, &ClientHandler::_processMedit
41 ate},
42                         {GameType::PLAYER_RESURRECT, &ClientHandler::_processResu
43 rrect},
44                         {GameType::PLAYER_SEND_MSG, &ClientHandler::_processMessa
45 ge},
46                         {GameType::PLAYER_HEAL, &ClientHandler::_processHeal},
47                         {GameType::PLAYER_REQUEST_INVENTORY_NAMES, &ClientHandler
48 ::_processInventoryNames}};
49 }
50
51 void ClientHandler::run() {
52     try {
53         uint32_t msgLength = 0;
54
55         while (!finished) {
56             buffer.clear();
57             socket.receive((char*)&(msgLength), sizeof(uint32_t));
58             msgLength = ntohs(msgLength);
59             buffer.resize(msgLength);
60             socket.receive(buffer.data(), msgLength);
61             _processClientAction(buffer);
62         }
63     } catch (std::exception& e) {
64         socket.close();
65         finished = true;
66     }
67 }

```

jul 21, 20 15:47

ClientHandler.cpp

Page 2/4

```

55     std::cerr << e.what() << std::endl;
56 }
57 }
58
59 void ClientHandler::sendGameUpdate() {
60     try {
61         const std::vector<char>& generalData = protocol.getGeneralData();
62         socket.send(generalData.data(), generalData.size());
63         std::vector<char> playerData = ServerProtocol::getPlayerData(player);
64         socket.send(playerData.data(), playerData.size());
65     } catch (std::exception& e) {
66         std::cerr << e.what() << std::endl;
67     }
68 }
69
70 bool ClientHandler::hasFinished() const {
71     return finished;
72 }
73
74 void ClientHandler::update() {
75     std::unique_lock<std::mutex> lk(m);
76     player.giveEventsToGame();
77 }
78
79 void ClientHandler::sendCurrentGameState(const std::vector<char>& gameState) {
80     try {
81         const std::vector<char>& mapInfo = protocol.getMapInfo();
82         socket.send(mapInfo.data(), mapInfo.size());
83         socket.send(gameState.data(), gameState.size());
84         std::vector<char> playerData = ServerProtocol::getPlayerData(player);
85         socket.send(playerData.data(), playerData.size());
86     } catch (std::exception& e) {
87         std::cerr << e.what() << std::endl;
88     }
89 }
90
91 void ClientHandler::forceFinish() {
92     socket.close();
93     finished = true;
94 }
95
96 void ClientHandler::setPlayerProxy(PlayerProxy^ _player) {
97     player = std::move(_player);
98 }
99
100 PlayerData ClientHandler::getPlayerData() const {
101     return player.getData();
102 }
103
104 //////////////////////////////////PRIVATE////////////////////////////////////
105
106 void ClientHandler::_processClientAction(std::vector<char>& data) {
107     offset = 0;
108     msgpack::type::tuple<GameType::PlayerEvent> event;
109     handler = msgpack::unpack(data.data(), data.size(), offset);
110     handler->convert(event);
111     std::unique_lock<std::mutex> lk(m);
112     try {
113         (this->*eventProcessors.at(std::get<0>(event)))(data);
114     } catch (std::out_of_range& e) {
115         std::cerr << "Received an unknown command from the client" << std::endl;
116     }
117 }
118
119 void ClientHandler::_processAttack(std::vector<char> &data) {

```

jul 21, 20 15:47

ClientHandler.cpp

Page 3/4

```

121 msgpack::type::tuple<int32_t, int32_t> attackInfo;
122 handler = msgpack::unpack(data.data(), data.size(), offset);
123 handler->convert(attackInfo);
124 player.attack({std::get<0>(attackInfo), std::get<1>(attackInfo)});
125 }
126
127 void ClientHandler::_processUseItem(std::vector<char> &data) {
128 msgpack::type::tuple<int32_t> itemPosition;
129 handler = msgpack::unpack(data.data(), data.size(), offset);
130 handler->convert(itemPosition);
131 player.useItem(std::get<0>(itemPosition));
132 }
133
134 void ClientHandler::_processUnequip(std::vector<char> &data) {
135 msgpack::type::tuple<GameType::EquipmentPlace> equipmentPlace;
136 handler = msgpack::unpack(data.data(), data.size(), offset);
137 handler->convert(equipmentPlace);
138 player.unequip(std::get<0>(equipmentPlace));
139 }
140
141 void ClientHandler::_processPickUp(std::vector<char> &data) {
142 player.pickUpItem();
143 }
144
145 void ClientHandler::_processDrop(std::vector<char> &data) {
146 msgpack::type::tuple<int32_t> itemPosition;
147 handler = msgpack::unpack(data.data(), data.size(), offset);
148 handler->convert(itemPosition);
149 player.dropItem(std::get<0>(itemPosition));
150 }
151
152 void ClientHandler::_processList(std::vector<char> &data) {
153 msgpack::type::tuple<int32_t, int32_t> listPosition;
154 handler = msgpack::unpack(data.data(), data.size(), offset);
155 handler->convert(listPosition);
156 player.listFrom({std::get<0>(listPosition), std::get<1>(listPosition)});
157 }
158
159 void ClientHandler::_processBuy(std::vector<char> &data) {
160 msgpack::type::tuple<std::string, int32_t, int32_t> buyArguments;
161 handler = msgpack::unpack(data.data(), data.size(), offset);
162 handler->convert(buyArguments);
163 player.buyFrom(std::move(std::get<0>(buyArguments)),
164 {std::get<1>(buyArguments), std::get<2>(buyArguments)});
165 }
166
167 void ClientHandler::_processSell(std::vector<char> &data) {
168 msgpack::type::tuple<std::string, int32_t, int32_t> sellArguments;
169 handler = msgpack::unpack(data.data(), data.size(), offset);
170 handler->convert(sellArguments);
171 player.sellTo(std::move(std::get<0>(sellArguments)),
172 {std::get<1>(sellArguments), std::get<2>(sellArguments)});
173 }
174
175 void ClientHandler::_processWithdraw(std::vector<char> &data) {
176 msgpack::type::tuple<std::string, int32_t, int32_t> sellArguments;
177 handler = msgpack::unpack(data.data(), data.size(), offset);
178 handler->convert(sellArguments);
179 player.withdrawFrom(std::move(std::get<0>(sellArguments)),
180 {std::get<1>(sellArguments), std::get<2>(sellArguments)});
181 }
182
183 void ClientHandler::_processDeposit(std::vector<char> &data) {
184 msgpack::type::tuple<std::string, int32_t, int32_t> depositArguments;
185 handler = msgpack::unpack(data.data(), data.size(), offset);
186 handler->convert(depositArguments);

```

jul 21, 20 15:47

ClientHandler.cpp

Page 4/4

```

187 player.depositTo(std::move(std::get<0>(depositArguments)),
188 {std::get<1>(depositArguments), std::get<2>(depositArgum
189 ents)});
190 }
191
192 void ClientHandler::_processMeditate(std::vector<char> &data) {
193 player.meditate();
194 }
195
196 void ClientHandler::_processResurrect(std::vector<char> &data) {
197 msgpack::type::tuple<int32_t, int32_t> resurrectArguments;
198 handler = msgpack::unpack(data.data(), data.size(), offset);
199 handler->convert(resurrectArguments);
200 player.reqesResurrect({std::get<0>(resurrectArguments),
201 std::get<1>(resurrectArguments)});
202 }
203
204 void ClientHandler::_processMessage(std::vector<char> &data) {
205 msgpack::type::tuple<std::string, std::string> messageArguments;
206 handler = msgpack::unpack(data.data(), data.size(), offset);
207 handler->convert(messageArguments);
208 player.messageOtherPlayer(std::move(std::get<0>(messageArguments)),
209 std::move(std::get<1>(messageArguments)));
210 }
211
212 void ClientHandler::_processHeal(std::vector<char> &data) {
213 msgpack::type::tuple<int32_t, int32_t> healArguments;
214 handler = msgpack::unpack(data.data(), data.size(), offset);
215 handler->convert(healArguments);
216 player.requestHeal({std::get<0>(healArguments), std::get<1>(healArguments)});
217 }
218
219 void ClientHandler::_processInventoryNames(std::vector<char> &data) {
220 player.getInventoryNames();
221 }
222
223
224 void ClientHandler::_processStartMoving(std::vector<char> &data) {
225 msgpack::type::tuple<GameType::Direction> moveInfo;
226 handler = msgpack::unpack(data.data(), data.size(), offset);
227 handler->convert(moveInfo);
228 player.startMoving(std::get<0>(moveInfo));
229 }
230
231
232 void ClientHandler::_processStopMoving(std::vector<char> &data) {
233 player.stopMoving();
234 }

```

jul 21, 20 15:47

ClientAcceptor.h

Page 1/1

```

1 //
2 // Created by marcos on 6/24/20.
3 //
4
5 #ifndef ARGENTUM_CLIENTACCEPTER_H
6 #define ARGENTUM_CLIENTACCEPTER_H
7
8 #include <list>
9 #include <memory>
10 #include "../libs/Thread.h"
11 #include <atomic>
12 #include <msgpack.hpp>
13 #include "../Entities/PlayerProxy.h"
14 #include "PlayerData.hpp"
15
16 class ServerProtocol;
17 class Socket;
18 class ClientHandler;
19 class ClientsMonitor;
20 class PlayerManager;
21
22 class ClientAcceptor : public Thread {
23 private:
24     ClientsMonitor& clients;
25     ServerProtocol& protocol;
26     Socket& serverSocket;
27     std::atomic<bool>& keepRunning;
28     PlayerManager& manager;
29     msgpack::object_handle handler;
30
31 //Clase que se encarga de aceptar los nuevos clientes que intentan conectarse
32 public:
33     ClientAcceptor(ClientsMonitor& _clients, ServerProtocol& _protocol,
34                   Socket& _serverSocket, std::atomic<bool>& _keepRunning,
35                   PlayerManager& _manager) :
36         clients(_clients), protocol(_protocol),
37         serverSocket(_serverSocket), keepRunning(_keepRunning),
38         manager(_manager) {}
39
40 //Comienza a ejecutar el thread aceptador
41 void run() override;
42
43 private:
44     PlayerData _receivePlayerInfo(Socket& clientSocket);
45     PlayerData _createPlayer(std::vector<char>& buffer, std::size_t& offset);
46     PlayerData _loadPlayer(std::vector<char>& buffer, std::size_t& offset);
47     void _acceptClients();
48     static bool _sendResponseToClient(Socket& clientSocket, GameType::Connection
49 Response status);
50 };
51
52 #endif //ARGENTUM_CLIENTACCEPTER_H

```

jul 21, 20 15:47

ClientAcceptor.cpp

Page 1/2

```

1 //
2 // Created by marcos on 6/24/20.
3 //
4
5 #include "ClientAcceptor.h"
6 #include "../libs/Socket.h"
7 #include "ClientHandler.h"
8 #include "ClientsMonitor.h"
9 #include <iostream>
10 #include "PlayerManager.h"
11 #include "../Exceptions/UnavailablePlayerException.h"
12 #include "../Exceptions/InexistentPlayerException.h"
13
14 MSGPACK_ADD_ENUM(GameType::PlayerEvent)
15 MSGPACK_ADD_ENUM(GameType::Class)
16 MSGPACK_ADD_ENUM(GameType::Race)
17
18 void ClientAcceptor::_acceptClients() {
19     GameType::ConnectionResponse status;
20     while (keepRunning) {
21         Socket clientSocket = serverSocket.accept();
22         try {
23             PlayerData playerData = _receivePlayerInfo(clientSocket);
24             status = GameType::ACCEPTED;
25             if (_sendResponseToClient(clientSocket, status)) {
26                 clients.pushToWaitingList(std::move(clientSocket), protocol, std
27 ::move(playerData));
28             }
29             catch(InexistentPlayerException& e) {
30                 status = GameType::INEXISTENT_PLAYER;
31                 _sendResponseToClient(clientSocket, status);
32             } catch (UnavailablePlayerException& e) {
33                 status = GameType::UNAVAILABLE_PLAYER;
34                 _sendResponseToClient(clientSocket, status);
35             } catch (std::exception& e) {
36                 std::cerr << e.what() << " in acceptor" << std::endl;
37                 status = GameType::UNKNOWN_SERVER_ERROR;
38                 _sendResponseToClient(clientSocket, status);
39             } catch(...) {
40                 std::cerr << "Unknown error while reading a client player information in acceptor!" << std
41 ::endl;
42                 status = GameType::UNKNOWN_SERVER_ERROR;
43                 _sendResponseToClient(clientSocket, status);
44             }
45         }
46
47 bool ClientAcceptor::_sendResponseToClient(Socket& clientSocket, GameType::Conne
48 ctionResponse status) {
49     status = static_cast<GameType::ConnectionResponse>(htonl(status));
50     try {
51         clientSocket.send(reinterpret_cast<char*>(&status), sizeof(status));
52     } catch(...) {
53         std::cerr << "Client disconnected suddenly in acceptor" << std::endl;
54         return false;
55     }
56     return true;
57
58 void ClientAcceptor::run() {
59     try {
60         _acceptClients();
61     } catch (std::exception& e) {
62         std::cerr << e.what() << " in acceptor socket" << std::endl;
63     } catch (...) {
64         std::cerr << "Unknown error in acceptor socket" << std::endl;
65     }
66 }

```


jul 21, 20 15:47

ClientAcceptor.cpp

Page 2/2

```

64     }
65     keepRunning = false;
66 }
67
68 PlayerData ClientAcceptor::_receivePlayerInfo(Socket& clientSocket) {
69     std::size_t offset = 0;
70     std::vector<char> buffer;
71     uint32_t msgLen;
72     clientSocket.receive(reinterpret_cast<char*>(&msgLen), sizeof(uint32_t));
73     msgLen = ntohl(msgLen);
74     buffer.clear();
75     buffer.resize(msgLen);
76     clientSocket.receive(buffer.data(), msgLen);
77     msgpack::type::tuple<GameType::PlayerEvent> creationID;
78     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
79     handler->convert(creationID);
80     if (std::get<0>(creationID) == GameType::CREATE_PLAYER) {
81         return _createPlayer(buffer, offset);
82     } else if (std::get<0>(creationID) == GameType::LOAD_PLAYER) {
83         return _loadPlayer(buffer, offset);
84     } else {
85         throw TPEException("Invalid load/create client messages!");
86     }
87 }
88
89 PlayerData ClientAcceptor::_createPlayer(std::vector<char>& buffer, std::size_t&
offset) {
90     msgpack::type::tuple<std::string, GameType::Race, GameType::Class> info;
91     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
92     handler->convert(info);
93     PlayerData playerData = {std::move(std::get<0>(info)),
94                             std::get<1>(info), std::get<2>(info)};
95     manager.storeNewPlayer(playerData);
96     return playerData;
97 }
98
99 PlayerData ClientAcceptor::_loadPlayer(std::vector<char>& buffer, std::size_t& o
ffset) {
100     msgpack::type::tuple<std::string> nickname;
101     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
102     handler->convert(nickname);
103     return manager.getSavedPlayerData(std::get<0>(nickname));
104 }

```

jul 21, 20 15:47

ArgentumServerSide.h

Page 1/1

```

1  #ifndef TP3TALLER_TP3SERVERSIDE_H
2  #define TP3TALLER_TP3SERVERSIDE_H
3
4  /*Esta es la clase principal del lado del ArgentumServer. Se encarga de validar
   los
5  * datos de ejecucion del ArgentumServer y, en caso de ser validos, instanciar a
   l
6  * ArgentumServer. Debe utilizarse esta clase para ejecutar el server */
7
8  class ArgentumServerSide {
9  public:
10     static int run(int argc, char** argv);
11 };
12
13
14 #endif //TP3TALLER_TP3SERVERSIDE_H

```

jul 21, 20 15:47

ArgentumServerSide.cpp

Page 1/1

```

1  #include <iostream>
2  #include "ArgentumServerSide.h"
3  #include "ArgentumServer.h"
4  #include "../Config/Configuration.h"
5
6  #define INVALID_ARGUMENTS_MESSAGE "Error: argumentos invalidos."
7  // #define ARGUMENT_AMOUNT 2
8  #define ERROR 1
9  #define SUCCESS 0
10 // #define PORT_ARG_INDEX 1
11 // #define MAP_PATH_ARG_INDEX 1
12
13 int ArgentumServerSide::run(int argc, char** argv) {
14     /*
15     if (argc != ARGUMENT_AMOUNT) {
16         std::cerr << INVALID_ARGUMENTS_MESSAGE << std::endl;
17         return ERROR;
18     }
19     */
20     try {
21         ArgentumServer server;
22         Configuration& config = Configuration::getInstance();
23         server.connect(config.configPort(), config.configMapPath());
24     } catch (std::exception& e) {
25         std::cerr << e.what() << std::endl;
26         return ERROR;
27     }
28     return SUCCESS;
29 }
```

jul 21, 20 15:47

ArgentumServer.h

Page 1/1

```

1  #ifndef TP3_SERVER_H
2  #define TP3_SERVER_H
3
4  /*Esta clase se encarga de manejar la
5   * logica de las conexiones pero no de la comunicacion con los clientes, de eso
6   * se encarga el Client Handler*/
7
8  #include "../libs/Socket.h"
9  #include <string>
10 #include <atomic>
11 #include <vector>
12 #include "ClientHandler.h"
13 #include <memory>
14 #include <utility>
15 #include "../Game/Game.h"
16 #include "ServerProtocol.h"
17 #include "ClientsMonitor.h"
18
19 class ArgentumServer {
20 private:
21     std::atomic<bool> keepRunning{true};
22     Socket socket;
23
24 public:
25     explicit ArgentumServer();
26     ArgentumServer(const ArgentumServer&) = delete; /*Borro los constructores po
27 r copia*/
28     ArgentumServer operator=(const ArgentumServer&) = delete;
29
30     /*Levanta el servidor en el puerto pedido en el constructor*/
31     /*Levanta el servidor en el puerto recibido*/
32     void connect(const std::string& _port, const std::string& mapFilePath);
33
34     /*Fuerza el cierre del servidor*/
35     void finish();
36
37 private:
38     void _execute(const std::string& mapFilePath);
39 };
40 #endif //TP3_SERVER_H
```

jul 21, 20 15:47

ArgentumServer.cpp

Page 1/2

```

1  #include <netdb.h>
2  #include "ArgentumServer.h"
3  #include "ServerMonitor.h"
4  #include "ClientAcceptor.h"
5  #include "../Config/MapFileReader.h"
6  #include "PlayerManager.h"
7  #include "../Config/Configuration.h"
8  #include <unistd.h>
9  #include <iostream>
10
11 const double FRAME_TIME = 1/60.f; /*ms que tarda en actualizarse el juego*/
12 const double TIME_FOR_CLIENTS_INITIALIZATION = 3; /*ms dejados para mandarle la
13 data inicial a los clientes
14 const double BACKUP_TIME = 5*60; /*5 minutos*/
15
16 using namespace std::chrono;
17
18 const int MAX_LISTENERS = 10;
19
20 void ArgentumServer::finish() {
21     keepRunning = false;
22     socket.close();
23 }
24
25 void ArgentumServer::connect(const std::string& _port, const std::string& mapFil
26 ePath) {
27     socket.bind(_port);
28     socket.maxListen(MAX_LISTENERS);
29     _execute(mapFilePath);
30 }
31
32 void ArgentumServer::_execute(const std::string& mapFilePath) {
33     Timer timeBetweenUpdates, timeBetweenBackups;
34     Game game((MapFileReader(mapFilePath)));
35     ServerProtocol protocol(game);
36     Configuration& config = Configuration::getInstance();
37     PlayerManager manager(game, protocol, config.configIndexPath(), config.conf
38 iSavePath());
39     ClientsMonitor clients(manager);
40     ServerMonitor monitor(*this);
41     monitor(); /*Espera la q para cerrar el server*/
42     ClientAcceptor acceptor(clients, protocol, socket, keepRunning, manager);
43     acceptor(); /*Acepta conexiones de clientes*/
44
45     timeBetweenBackups.start();
46
47     try {
48         double lastFrameTime = 0;
49         double lastBackupTime;
50         while (keepRunning) {
51             timeBetweenUpdates.start();
52             clients.removeDisconnectedClients(protocol);
53             clients.mergeClientsEvents();
54             game.update(lastFrameTime, protocol);
55             protocol.buildGeneralDataBuffer();
56             clients.sendGameUpdate();
57
58             lastFrameTime = timeBetweenUpdates.getTime();
59             lastBackupTime = timeBetweenBackups.getTime();
60             if (lastBackupTime / 1000 >= BACKUP_TIME) {
61                 clients.backup();
62                 timeBetweenBackups.start();
63                 std::cout << "Backuper players. Next backup in 5 minutes" << std::endl;
64             }
65             if (clients.hasWaitingClients() ^
66                 (FRAME_TIME*1000 - lastFrameTime) > TIME_FOR_CLIENTS_INITIALIZAT

```

jul 21, 20 15:47

ArgentumServer.cpp

Page 2/2

```

64     ION) {
65         clients.mergeWaitingClients(game, protocol);
66     }
67     lastFrameTime = timeBetweenUpdates.getTime();
68     if (lastFrameTime < FRAME_TIME*1000) {
69         usleep((FRAME_TIME*1000 - lastFrameTime) * 1000);
70         lastFrameTime = FRAME_TIME*1000;
71     }
72 }
73
74 } catch (std::exception& e) {
75     std::cerr << e.what() << std::endl;
76 } catch (...) {
77     std::cerr << "Unknown error in Main Game Loop!" << std::endl;
78 }
79
80 try {
81     if (monitor.closeRequest()) {
82         clients.backup();
83     }
84 } catch (std::exception& e) {
85     std::cerr << e.what() << "while backing up players!" << std::endl;
86 } catch (...) {
87     std::cerr << "Unknown error while backing up players!" << std::endl;
88 }
89
90 finish();
91 monitor.join(); /*Joineamos los threads*/
92 clients.join();
93 acceptor.join();
94 }
95
96 ArgentumServer::ArgentumServer() = default;

```

jul 21, 20 15:47

SaveFileManager.h

Page 1/1

```

1  //
2  // Created by marcos on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_SAVEFILEMANAGER_H
6  #define ARGENTUM_SAVEFILEMANAGER_H
7
8  #include "PlayerIndexFile.h"
9  #include "PlayerSaveFile.h"
10 #include <mutex>
11
12 /*Esta clase se encarga de administrar la persistencia del servidor. Maneja los
13  * 2 archivos (indice y save file)*/
14
15 class SaveFileManager {
16 private:
17     PlayerIndexFile indexFile;
18     PlayerSaveFile saveFile;
19     std::mutex m;
20
21 public:
22     SaveFileManager(const std::string& indexPath, const std::string& savePath) :
23         indexFile(indexPath), saveFile(savePath) {}
24
25     /*Retorna la data almacenada del player. Si el player no existe tira excepti
26 on*/
27     PlayerData getPlayerData(const std::string& playerNickname);
28
29     /*Almacena la data del player recibida en el correspondiente en el archivo.
30  * Si el player no existia en el archivo tira exception*/
31     void storeOldPlayer(const PlayerData& data);
32
33     /*Almacena la data del player recibida, agregando la entrada correspondiente
34  * en ambos archivos. Si el player ya existia tira exception*/
35     void storeNewPlayer(const PlayerData& data);
36
37 };
38 #endif //ARGENTUM_SAVEFILEMANAGER_H

```

jul 21, 20 15:47

SaveFileManager.cpp

Page 1/1

```

1  //
2  // Created by marcos on 9/7/20.
3  //
4
5  #include "SaveFileManager.h"
6  #include "../Exceptions/UnavailablePlayerException.h"
7
8  PlayerData SaveFileManager::getPlayerData(const std::string &playerNickname) {
9      std::lock_guard<std::mutex> l(m);
10     PlayerFilePosition filePosition = indexFile.getPlayerPosition(playerNickname
11 );
12     return saveFile.getPlayerData(playerNickname, filePosition);
13 }
14
15 void SaveFileManager::storeNewPlayer(const PlayerData &data) {
16     std::lock_guard<std::mutex> l(m);
17     if (indexFile.playerExists(data.nickname)) {
18         throw UnavailablePlayerException();
19     }
20     PlayerFilePosition filePosition = saveFile.storePlayerData(data);
21     indexFile.storeNewPlayer(data.nickname, filePosition);
22 }
23
24 void SaveFileManager::storeOldPlayer(const PlayerData &data) {
25     std::lock_guard<std::mutex> l(m);
26     PlayerFilePosition filePosition = indexFile.getPlayerPosition(data.nickname)
27 ;
28     filePosition = saveFile.storePlayerData(data, filePosition.offset);
29     indexFile.storeOldPlayer(data.nickname, filePosition);
30 }

```

jul 21, 20 15:47

PlayerSaveFile.h

Page 1/1

```

1  //
2  // Created by marcos on 7/8/20.
3  //
4
5  #ifndef ARGENTUM_PLAYERSAVEFILE_H
6  #define ARGENTUM_PLAYERSAVEFILE_H
7
8  #include <fstream>
9  #include "PlayerFilePosition.h"
10 #include "../Server/PlayerData.hpp"
11 #include <vector>
12
13 /*Esta clase es la que maneja el archivo que contiene los datos del player
14  * guardado*/
15 class PlayerSaveFile {
16 private:
17     std::fstream saveFile;
18     msgpack::object_handle handler;
19     std::size_t readData{0};
20
21 public:
22     explicit PlayerSaveFile(const std::string& filePath);
23
24     /*Retorna la data guardada del player (stats, inventario, banker items, etc)
25     */
26     PlayerData getPlayerData(const std::string& playerNickname,
27                             PlayerFilePosition filePosition);
28
29     /*Almacena la informacion del player provisto en la posicion especificada
30     * en el offset*/
31     PlayerFilePosition storePlayerData(const PlayerData& playerData,
32                                       int32_t fileOffset);
33
34     /*Sobrecarga para storePlayerData, esta funcion se usa para almacenar un nue
35     vo
36     * player en el archivo (no recibe offset porque siempre ira al final del ar
37     chivo*/
38     PlayerFilePosition storePlayerData(const PlayerData& playerData);
39
40 private:
41     static void _packPlayerType(std::stringstream& dataToStore, const PlayerData
42     & playerData);
43     static void _packPlayerGeneralStats(std::stringstream& dataToStore,
44                                         const PlayerData& playerData);
45     static void _packPlayerInventory(std::stringstream& dataToStore,
46                                     const PlayerData& playerData);
47     void _loadPlayerType(PlayerData& playerData,
48                         std::vector<char>& playerDataBuffer);
49     void _loadPlayerGeneralStats(PlayerData& playerData,
50                                 std::vector<char>& playerDataBu
51 ffer);
52     void _loadPlayerInventory(PlayerData& playerData, std::vector<char>& playerD
53 ataBuffer);
54     static void _packBankItems(std::stringstream& dataToStore, const PlayerData&
55 playerData);
56     void _loadPlayerBank(PlayerData& playerData, std::vector<char>& playerDataBu
57 ffer);
58
59 };
60
61 #endif //ARGENTUM_PLAYERSAVEFILE_H

```

jul 21, 20 15:47

PlayerSaveFile.cpp

Page 1/4

```

1  //
2  // Created by marcos on 7/8/20.
3  //
4
5  #include "PlayerSaveFile.h"
6  #include <iostream>
7  #include "../libs/TPEException.h"
8
9  MSGPACK_ADD_ENUM(GameType::Race)
10 MSGPACK_ADD_ENUM(GameType::Class)
11 MSGPACK_ADD_ENUM(GameType::ItemType)
12 MSGPACK_ADD_ENUM(GameType::EquipmentPlace)
13
14 PlayerSaveFile::PlayerSaveFile(const std::string& filePath) {
15     saveFile.open(filePath, std::ios::in | std::ios::out | std::ios::binary);
16     if (!saveFile.is_open()) {
17         std::cout << "Could not find a Save File with the provided name."
18                 << " Creating one now." << std::endl;
19         std::ofstream newSaveFile(filePath);
20         newSaveFile.close();
21         saveFile.open(filePath, std::ios::in | std::ios::out | std::ios::binary)
22     ;
23     }
24
25     PlayerData PlayerSaveFile::getPlayerData(const std::string& playerNickname,
26                                             PlayerFilePosition filePosition) {
27         saveFile.clear();
28         saveFile.seekg(filePosition.offset, std::ios_base::beg);
29         readData = 0;
30         std::vector<char> playerDataBuffer(filePosition.length);
31         saveFile.read(playerDataBuffer.data(), filePosition.length);
32         PlayerData playerData;
33         _loadPlayerType(playerData, playerDataBuffer);
34         if (playerData.nickname != playerNickname) {
35             throw TPEException("Stored player's nickname doesnt match the one provided!");
36         }
37         _loadPlayerGeneralStats(playerData, playerDataBuffer);
38         _loadPlayerInventory(playerData, playerDataBuffer);
39         _loadPlayerBank(playerData, playerDataBuffer);
40         return playerData;
41     }
42
43     void PlayerSaveFile::_loadPlayerBank(PlayerData& playerData,
44                                         std::vector<char>& playerDataBuffer) {
45
46         for (auto & currItem : playerData.bankerItems) {
47             msgpack::type::tuple<GameType::ItemType, int32_t> item;
48             handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), re
49 adData);
50             handler->convert(item);
51             currItem = item;
52         }
53         msgpack::type::tuple<int32_t> bankGold;
54         handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
55 ta);
56         handler->convert(bankGold);
57         playerData.bankerGold = std::get<0>(bankGold);
58     }
59
60     void PlayerSaveFile::_loadPlayerInventory(PlayerData& playerData,
61                                             std::vector<char>& playerDataBuffer
62 ) {
63
64         for (auto & currItem : playerData.inventory) {
65             msgpack::type::tuple<GameType::ItemType, int32_t> item;

```

jul 21, 20 15:47

PlayerSaveFile.cpp

Page 2/4

```

63     handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), re
adData);
64     handler->convert(item);
65     currItem = item;
66 }
67 msgpack::type::tuple<int32_t> equipment;
68 handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
69 handler->convert(equipment);
70 playerData.equipment.at(GameType::EQUIPMENT_PLACE_HEAD) = std::get<0>(equipm
ent);
71 handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
72 handler->convert(equipment);
73 playerData.equipment.at(GameType::EQUIPMENT_PLACE_CHEST) = std::get<0>(equip
ment);
74 handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
75 handler->convert(equipment);
76 playerData.equipment.at(GameType::EQUIPMENT_PLACE_SHIELD) = std::get<0>(equi
pment);
77 handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
78 handler->convert(equipment);
79 playerData.equipment.at(GameType::EQUIPMENT_PLACE_WEAPON) = std::get<0>(equi
pment);
80 }
81
82 void PlayerSaveFile::_loadPlayerGeneralStats(PlayerData& playerData,
83     std::vector<char>& playerDataBuffer) {
84     msgpack::type::tuple<int32_t, int32_t, int32_t> generalStats;
85     handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
86     handler->convert(generalStats);
87     playerData.level = std::get<0>(generalStats);
88     playerData.experience = std::get<1>(generalStats);
89     playerData.gold = std::get<2>(generalStats);
90     msgpack::type::tuple<int32_t, int32_t, int32_t> stats;
91     handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
92     handler->convert(stats);
93     playerData.constitution = std::get<0>(stats);
94     playerData.strength = std::get<1>(stats);
95     playerData.agility = std::get<2>(stats);
96     playerData.intelligence = std::get<3>(stats);
97 }
98
99 void PlayerSaveFile::_loadPlayerType(PlayerData& playerData,
100     std::vector<char>& playerDataBuffer) {
101     msgpack::type::tuple<bool> isNewPlayer;
102     handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
103     handler->convert(isNewPlayer);
104     playerData.isNewPlayer = std::get<0>(isNewPlayer);
105     msgpack::type::tuple<std::string, GameType::Race, GameType::Class> playerTyp
e;
106     handler = msgpack::unpack(playerDataBuffer.data(), playerData.size(), readDa
ta);
107     handler->convert(playerType);
108     playerData.nickname = std::move(std::get<0>(playerType));
109     playerData.pRace = std::get<1>(playerType);
110     playerData.pClass = std::get<2>(playerType);
111 }
112
113 PlayerFilePosition PlayerSaveFile::storePlayerData(const PlayerData& playerData,
114     int32_t fileOffset) {

```

jul 21, 20 15:47

PlayerSaveFile.cpp

Page 3/4

```

115     saveFile.clear();
116     PlayerFilePosition playerPosition{};
117     playerPosition.offset = fileOffset;
118     std::stringstream dataToStore;
119     _packPlayerType(dataToStore, playerData);
120     _packPlayerGeneralStats(dataToStore, playerData);
121     _packPlayerInventory(dataToStore, playerData);
122     _packBankItems(dataToStore, playerData);
123     std::string strDataToStore = dataToStore.str();
124     playerPosition.length = strDataToStore.size();
125     saveFile.seekp(fileOffset, std::ios_base::beg);
126     saveFile.write(strDataToStore.data(), playerPosition.length);
127     int32_t paddingSize = (playerData.size() - playerPosition.length);
128     std::vector<char> paddingBuffer(paddingSize, 0);
129     saveFile.write(paddingBuffer.data(), paddingSize);
130     saveFile.sync();
131     return playerPosition;
132 }
133
134 void PlayerSaveFile::_packBankItems(std::stringstream& dataToStore,
135     const PlayerData& playerData) {
136     for (auto & currItem : playerData.bankerItems) {
137         msgpack::type::tuple<GameType::ItemType, int32_t> item(currItem);
138         msgpack::pack(dataToStore, item);
139     }
140     msgpack::type::tuple<int32_t> gold(playerData.bankerGold);
141     msgpack::pack(dataToStore, gold);
142 }
143
144 void PlayerSaveFile::_packPlayerType(std::stringstream& dataToStore,
145     const PlayerData& playerData) {
146     msgpack::type::tuple<bool> isNewPlayer(playerData.isNewPlayer);
147     msgpack::pack(dataToStore, isNewPlayer);
148     msgpack::type::tuple<std::string, GameType::Race, GameType::Class> playerTyp
e(
149         playerData.nickname, playerData.pRace, playerData.pClass
150     );
151     msgpack::pack(dataToStore, playerType);
152 }
153 void PlayerSaveFile::_packPlayerGeneralStats(std::stringstream& dataToStore,
154     const PlayerData& playerData) {
155     msgpack::type::tuple<int32_t, int32_t, int32_t> generalStats(playerData.level
156     ,
157         playerData.experience, playerData.gold);
158     msgpack::pack(dataToStore, generalStats);
159     msgpack::type::tuple<int32_t, int32_t, int32_t, int32_t> stats(playerData.co
nstitution,
160         playerData.strength, playerData.agility, playerD
161     ata.intelligence);
162     msgpack::pack(dataToStore, stats);
163 }
164 void PlayerSaveFile::_packPlayerInventory(std::stringstream& dataToStore,
165     const PlayerData& playerData) {
166     for (auto & currItem : playerData.inventory) {
167         msgpack::type::tuple<GameType::ItemType, int32_t> item(currItem);
168         msgpack::pack(dataToStore, item);
169     }
170     msgpack::type::tuple<int32_t> helmet(playerData.equipment.at(GameType::EQUIP
MENT_PLACE_HEAD));
171     msgpack::pack(dataToStore, helmet);
172     msgpack::type::tuple<int32_t> chest(playerData.equipment.at(GameType::EQUIPM
ENT_PLACE_CHEST));
173     msgpack::pack(dataToStore, chest);
174     msgpack::type::tuple<int32_t> shield(playerData.equipment.at(GameType::EQUIP

```

jul 21, 20 15:47

PlayerSaveFile.cpp

Page 4/4

```

    MENT_PLACE_SHIELD));
    msgpack::pack(dataToStore, shield);
174     msgpack::type::tuple<int32_t> weapon(playerData.equipment.at(GameType::EQUIP
175     MENT_PLACE_WEAPON));
    msgpack::pack(dataToStore, weapon);
176 }
177 }
178
179 PlayerFilePosition PlayerSaveFile::storePlayerData(const PlayerData &playerData)
{
180     saveFile.seekp(0, std::ios_base::end);
181     return storePlayerData(playerData, saveFile.tellp());
182 }

```

jul 21, 20 15:47

PlayerIndexFile.h

Page 1/1

```

1  //
2  // Created by marcos on 7/8/20.
3  //
4
5  #ifndef ARGENTUM_PLAYERINDEXFILE_H
6  #define ARGENTUM_PLAYERINDEXFILE_H
7
8  #include <fstream>
9  #include <unordered_map>
10 #include "PlayerFilePosition.h"
11
12 /*Esta clase maneja el archivo de indice de jugadores, el cual contiene
13  * el nombre del jugador, el offset donde arranca su informacion y la cantidad d
14  e
15  * bytes que su informacion ocupa (para saber cuanto leer con msgpack)*/
16 class PlayerIndexFile {
17 private:
18     std::fstream indexFile;
19     std::unordered_map<std::string, PlayerFilePosition> players;
20     std::unordered_map<std::string, int32_t> indexPlayersPosition; /*Guardo la p
21 osicion en el indice del largo en bytes*/
22                                     /*de la info
23 del player para poder acceder rapido cada vez que cambie*/
24
25 public:
26     explicit PlayerIndexFile(const std::string& filePath);
27
28     /*Almacena los datos actualizados de un player player ya estaba almacenado e
29 n el archivo*/
30     void storeOldPlayer(const std::string& playerNickname, PlayerFilePosition fi
31 lePosition);
32
33     /*Almacena en el archivo los datos de un player que acaba de ser creado*/
34     void storeNewPlayer(const std::string& playerNickname, PlayerFilePosition fi
35 lePosition);
36
37     /*Retorna la posicion del player en el archivo que contiene todos sus datos*/
38     /
39     PlayerFilePosition getPlayerPosition(const std::string& nickname);
40
41     /*Retorna true si el player existe en el archivo de datos, false en caso con
42 trario*/
43     bool playerExists(const std::string& nickname) const;
44
45 private:
46     void _loadFileData();
47 };
48
49 #endif //ARGENTUM_PLAYERINDEXFILE_H

```

jul 21, 20 15:47

PlayerIndexFile.cpp

Page 1/2

```

1  //
2  // Created by marcos on 7/8/20.
3  //
4
5  #include "PlayerIndexFile.h"
6  #include <iostream>
7  #include <arpa/inet.h>
8  #include <vector>
9  #include "../libs/TPEException.h"
10 #include "../Exceptions/InexistentPlayerException.h"
11
12 PlayerIndexFile::PlayerIndexFile(const std::string& filePath) {
13     indexFile.open(filePath, std::ios::in | std::ios::out | std::ios::binary);
14     if (!indexFile.is_open()) {
15         std::cout << "Could not find an Index File with the provided name."
16             " Creating one now." << std::endl;
17         std::ofstream newIndexFile(filePath);
18         newIndexFile.close();
19         indexFile.open(filePath, std::ios::in | std::ios::out | std::ios::binary);
20     } else {
21         _loadFileData();
22     }
23 }
24
25 void PlayerIndexFile::_loadFileData() {
26     PlayerFilePosition offData{};
27     uint32_t nicknameSize = 0;
28     indexFile.seekg(0, std::ios_base::beg);
29     while (indexFile.peek() != std::fstream::traits_type::eof() && !indexFile.eof()) {
30         indexFile.read(reinterpret_cast<char*>(&nicknameSize), sizeof(nicknameSize));
31         nicknameSize = ntohl(nicknameSize);
32         std::vector<char> playerNickname(nicknameSize);
33         indexFile.read(playerNickname.data(), nicknameSize);
34         indexFile.read(reinterpret_cast<char*>(&offData.offset), sizeof(offData.offset));
35         indexPlayersPosition.emplace(playerNickname.data(), indexFile.tellg());
36         indexFile.read(reinterpret_cast<char*>(&offData.length), sizeof(offData.length));
37         offData.offset = ntohl(offData.offset);
38         offData.length = ntohl(offData.length);
39         players.emplace(playerNickname.data(), offData);
40     }
41 }
42
43 void PlayerIndexFile::storeOldPlayer(const std::string& playerNickname, PlayerFilePosition filePosition) {
44     indexFile.clear();
45     if (players.count(playerNickname) == 1) {
46         players.at(playerNickname) = filePosition;
47         filePosition = {htonl(filePosition.offset), htonl(filePosition.length)};
48         indexFile.seekp(indexPlayersPosition.at(playerNickname), std::ios_base::beg);
49         indexFile.write(reinterpret_cast<char*>(&filePosition.length), sizeof(filePosition.length));
50         indexFile.sync();
51     } else {
52         throw TPEException("Tried to store a logged in player that didnt exist!");
53     }
54 }
55
56 PlayerFilePosition PlayerIndexFile::getPlayerPosition(const std::string& nickname) {
57     if (playerExists(nickname)) {

```

jul 21, 20 15:47

PlayerIndexFile.cpp

Page 2/2

```

58         return players.at(nickname);
59     }
60     throw InexistentPlayerException();
61 }
62
63 void PlayerIndexFile::storeNewPlayer(const std::string& playerNickname,
64     PlayerFilePosition filePosition) {
65
66     players.emplace(playerNickname, filePosition);
67     indexFile.clear();
68     indexFile.seekp(0, std::ios_base::end);
69     filePosition = {htonl(filePosition.offset), htonl(filePosition.length)};
70     uint32_t nameLength = playerNickname.size() + 1;
71     nameLength = htonl(nameLength);
72     indexFile.write(reinterpret_cast<char*>(&nameLength), sizeof(nameLength));
73     indexFile.write(playerNickname.data(), playerNickname.size() + 1);
74     indexFile.write(reinterpret_cast<char*>(&filePosition.offset), sizeof(filePosition.offset));
75     indexPlayersPosition.emplace(playerNickname, indexFile.tellp());
76     indexFile.write(reinterpret_cast<char*>(&filePosition.length), sizeof(filePosition.length));
77     indexFile.sync();
78 }
79
80 bool PlayerIndexFile::playerExists(const std::string& nickname) const {
81     return (players.count(nickname) == 1);
82 }

```


jul 21, 20 15:47

PlayerFilePosition.h

Page 1/1

```

1 //
2 // Created by marcos on 7/8/20.
3 //
4
5 #ifndef ARGENTUM_PLAYERFILEPOSITION_H
6 #define ARGENTUM_PLAYERFILEPOSITION_H
7
8 /*Este struct se usa para saber el offset de un player en el archivo (donde comi
enza)
9  * y la longitud que ocupan sus datos en este*/
10
11 struct PlayerFilePosition {
12     uint32_t offset;
13     uint32_t length;
14 };
15
16 #endif //ARGENTUM_PLAYERFILEPOSITION_H

```

jul 21, 20 15:47

Tile.h

Page 1/2

```

1 //
2 // Created by agustin on 6/6/20.
3 //
4
5 #ifndef ARGENTUM_TILE_H
6 #define ARGENTUM_TILE_H
7
8 #include <memory>
9 #include <list>
10 #include "../libs/GameEnums.h"
11 #include "../Entities/Entity.h"
12 #include "../Items/ItemData.h"
13
14 class MapTests;
15 class Item;
16 struct AttackResult;
17 struct ProductData;
18 class Player;
19
20 class Tile {
21 private:
22     std::shared_ptr<Entity> entity;
23     std::list<std::shared_ptr<Item>> items;
24     bool isOccupable{};
25     bool isFromCity;
26     GameType::FloorType floor;
27     GameType::Structure structure{GameType::Structure::TREE};
28
29     friend MapTests;
30
31 private:
32     void _storeItemsNames(Player& player);
33
34 public:
35     explicit Tile(bool isOccupable, bool isFromCity, GameType::FloorType floor,
36                 GameType::Structure structure, std::shared_ptr<Entity>^ initi
alEntity);
37
38     //El tile se queda con la entity de other y setea la de other en nullptr,
39     //actualizando tambien el estado de si es ocupable o no en ambos tiles
40     void moveEntity(Tile& otherTile, Coordinate position);
41
42     //Intenta agregar la entity al tile
43     //Si la posicion es ocupable entonces se apropia del puntero
44     void addEntity(std::shared_ptr<Entity>^ received_entity);
45
46     //Elimina la entity guardada, habilita la ocupacion del tile por otra
47     //entity
48     void removeEntity();
49
50     //Intenta agregar el item al tile, sumandolo a los items ya guardados
51     void addItem(std::shared_ptr<Item>^ received_item);
52
53     //Agrega los items recibidos en la lista a los items que contiene el tile
54     void addItem(std::list<std::shared_ptr<Item>>^ _items);
55
56     //Elimina uno de los items que se encuentran en el tile y lo retorna}
57     //Si no hay ningun item retorna nullptr
58     std::shared_ptr<Item> removeItem();
59
60     //Ataca la entidad que se encuentre guardada en el mapa
61     //Retorna la cantidad de daño que recibio la entidad atacada, si no hay
62     //una entidad retorna 0, el booleano indica si se realizo o no el ataque al
tile,
63     //valiendo true si se realizo, false en otro caso (que no haya un entity no
hace

```

jul 21, 20 15:47

Tile.h

Page 2/2

```

64 //necesariamente que sea false)
65 std::pair<AttackResult, bool> attacked(int damage, unsigned int level, bool
    isAPlayer);
66
67 //Retorna true si almacena un entity que es un target de un monster
68 bool hasMonsterTarget() const;
69
70 //Indica si el tile puede ser ocupado por una entity
71 //Retorna true si puede, false en otro caso
72 bool isAvailable() const;
73
74 //Delega el comportamiento a la entity que guarda, si es que guarda una
75 //void list(Player& player, std::list<ProductData>& products);
76 void list(Player& player);
77
78 //Delega el comportamiento a la entity que guarda, si es que guarda una
79 void withdraw(Player& player, const std::string& itemName);
80
81 //Delega el comportamiento a la entity que guarda, si es que guarda una
82 void deposit(Player& player, const std::string& itemName);
83
84 //Delega el comportamiento a la entity que guarda, si es que guarda una
85 void buy(Player& player, const std::string& itemName);
86
87 //Delega el comportamiento a la entity que guarda, si es que guarda una
88 void sell(Player& player, const std::string& itemName);
89
90 //Retorna si es de una city
91 bool isInCity() const;
92
93 //Guarda en el buffer el entity almacenado en el tile, junto con el tipo de
94 //piso y la estructura almacenadas
95 void operator>>(std::stringstream & mapBuffer) const;
96
97 //Retorna un puntero al item que se debe mostrar en el mapa
98 const Item* peekShowedItemData();
99
100 //Le pide al entity guardado en el tile que restaure la vida y el mana del
101 //player recibido, si no hay un entity entonces no hace nada
102 void requestRestore(Player& player);
103 };
104
105
106 #endif //ARGENTUM_TILE_H

```

jul 21, 20 15:47

Tile.cpp

Page 1/3

```

1 //
2 // Created by agustin on 6/6/20.
3 //
4
5 #include "Tile.h"
6 #include <memory>
7 #include "../Entities/AttackResult.h"
8 #include "../libs/TPEException.h"
9 #include "../Items/Item.h"
10 #include "../Entities/Player.h"
11 #include <msgpack.hpp>
12
13 MSGPACK_ADD_ENUM(GameType::FloorType)
14 MSGPACK_ADD_ENUM(GameType::Entity)
15 MSGPACK_ADD_ENUM(GameType::Structure)
16
17 #define NO_ITEMS_MESSAGE "There are no entities or items on this tile\n"
18 #define ITEMS_MESSAGE "The stored items are:\n"
19
20 //////////////////////////////////PUBLIC////////////////////////////////////
21
22 Tile::Tile(bool _isOccupable, bool _isFromCity, GameType::FloorType _floor, Game
    Type::Structure _structure,
23     std::shared_ptr<Entity>^ initialEntity): entity(nullptr) {
24     floor = _floor;
25     structure = _structure;
26     isFromCity = _isFromCity;
27     isOccupable = _isOccupable;
28
29     if (isOccupable ^ initialEntity) {
30         entity = std::move(initialEntity);
31         isOccupable = false;
32     }
33 }
34
35 void Tile::moveEntity(Tile& otherTile, Coordinate position) {
36     this->entity = std::move(otherTile.entity);
37     otherTile.entity = nullptr;
38     entity->move(position);
39     isOccupable = false;
40     otherTile.isOccupable = true;
41 }
42
43 void Tile::addEntity(std::shared_ptr<Entity>^ received_entity) {
44     if (isOccupable) {
45         entity = std::move(received_entity);
46         isOccupable = false;
47     } else {
48         throw TPEException("Tried to add an entity to a non occupable tile!");
49     }
50 }
51
52 void Tile::removeEntity() {
53     if (entity) {
54         entity.reset();
55         isOccupable = true;
56     }
57 }
58
59 void Tile::addItem(std::shared_ptr<Item>^ received_item) {
60     if (received_item) {
61         items.push_back(std::move(received_item));
62     }
63 }
64

```

jul 21, 20 15:47

Tile.cpp

Page 2/3

```

65  std::shared_ptr<Item> Tile::removeItem() {
66      if (items.empty()) {
67          return nullptr;
68      }
69      std::shared_ptr<Item> return_item = std::move(items.back());
70      items.pop_back();
71      return return_item;
72  }
73
74  std::pair<AttackResult, bool> Tile::attacked(int damage, unsigned int level, bool
  isAPlayer) {
75      if (isFromCity) {
76          return {{0, 0, "You can't attack a tile inside a city\n"}, false};
77      }
78      if (!entity) {
79          return {{0, 0, ""}, true};
80      }
81      return {entity->attacked(damage, level, isAPlayer), true};
82  }
83
84  bool Tile::hasMonsterTarget() const {
85      if (entity) {
86          return entity->isMonsterTarget();
87      }
88      return false;
89  }
90
91  bool Tile::isAvailable() const {
92      return isOccupable;
93  }
94
95  void Tile::list(Player &player) {
96      if (entity) {
97          entity->list(player);
98      } else {
99          _storeItemsNames(player);
100      }
101  }
102
103
104  void Tile::withdraw(Player &player, const std::string &itemName) {
105      if (entity) {
106          entity->withdraw(player, itemName);
107      }
108  }
109
110  void Tile::deposit(Player &player, const std::string &itemName) {
111      if (entity) {
112          entity->deposit(player, itemName);
113      }
114  }
115
116  void Tile::buy(Player &player, const std::string &itemName) {
117      if (entity) {
118          entity->buy(player, itemName);
119      }
120  }
121
122  void Tile::sell(Player &player, const std::string &itemName) {
123      if (entity) {
124          entity->sell(player, itemName);
125      }
126  }
127
128  void Tile::addItem(std::list<std::shared_ptr<Item>> &_items) {
129      for (auto & item : _items) {

```

jul 21, 20 15:47

Tile.cpp

Page 3/3

```

130      if (item) {
131          items.push_back(std::move(item));
132      }
133  }
134  }
135
136  bool Tile::isInCity() const {
137      return isFromCity;
138  }
139
140  void Tile::operator>>(std::stringstream &mapBuffer) const {
141      GameType::Entity entityType = GameType::Entity::NOTHING;
142      std::string nickname = "";
143      if (entity) {
144          entityType = entity->getType();
145          nickname = entity->getNickname();
146      }
147      /*de izquierda a derecha es el tipo de piso, tipo de estructura y citizen*/
148      msgpack::type::tuple<GameType::FloorType, GameType::Structure,
149          GameType::Entity, std::string> tileInfo(floor, structure, entityType
150      , nickname);
151      msgpack::pack(mapBuffer, tileInfo);
152  }
153
154  const Item* Tile::peekShowedItemData() {
155      if (items.empty()) {
156          return nullptr;
157      }
158      return items.back().get();
159  }
160
161  void Tile::requestRestore(Player& player) {
162      if (entity) {
163          entity->requestHeal(player);
164      }
165  }
166
167  //////////////////////////////////PRIVATE////////////////////////////////////
168
169  void Tile::_storeItemsNames(Player& player) {
170      if (!items.empty()) {
171          player.addMessage(ITEMS_MESSAGE);
172          for (const auto & item: items) {
173              player.addMessage(item->getName() + "\n");
174          }
175      }
176  }

```

jul 21, 20 15:47

PointAndDistance.h

Page 1/1

```

1  //
2  // Created by agustin on 10/6/20.
3  //
4
5  #ifndef ARGENTUM_POINTANDDISTANCE_H
6  #define ARGENTUM_POINTANDDISTANCE_H
7
8  #include "Coordinate.h"
9
10 //Este struct es auxiliar, solo deberia ser usado por el mapa para pathfinding,
11 //se declara en el header por tener que declarar la funcion privada
12 struct PointAndDistance {
13     Coordinate point;
14     unsigned int distance;
15 };
16
17
18 #endif //ARGENTUM_POINTANDDISTANCE_H

```

jul 21, 20 15:47

Map.h

Page 1/3

```

1  //
2  // Created by agustin on 7/6/20.
3  //
4
5  #ifndef ARGENTUM_MAP_H
6  #define ARGENTUM_MAP_H
7
8  #include <vector>
9  #include <list>
10 #include <unordered_map>
11 #include <queue>
12 #include <memory>
13 #include "InverseCoordinateDistance.h"
14 #include "Tile.h"
15
16 struct AttackResult;
17 class Item;
18 class Entity;
19 class Monster;
20 class MapFileReader;
21 class EntityTests;
22 class MapTests;
23
24 class Map {
25 private:
26     std::vector<std::vector<Tile>> tiles;
27
28     friend EntityTests;
29     friend MapTests;
30
31 private:
32     void _storeAdjacentPositions(PointAndDistance refference,
33                                std::unordered_map<Coordinate, unsigned int>& distances,
34                                std::unordered_map<Coordinate, Coordinate>& parentsAndChilds,
35                                std::priority_queue<PointAndDistance, std::vector<PointAndDistance>,
36                                                InverseCoordinateDistance>& nodes, Coordinate de
37                                stination) const;
38     static unsigned int _getDistance(Coordinate a, Coordinate b);
39     bool _isCoordinateValid(Coordinate coordinate) const;
40     Coordinate _getValidCoordinate(Coordinate coordinate) const;
41     static void _storePath(Coordinate initialPosition, Coordinate desiredPosition, const std::unordered_map<Coordinate,
42                                Coordinate>& parentsAndChilds, std::list<Coordinate>& path);
43     void _buildSearchRegion(Coordinate center, unsigned int range, Coordinate& topLeft, Coordinate& bottomRight) const;
44     static bool _areCoordinatesEqual(Coordinate a, Coordinate b);
45     static void _initializeConstructorMaps(std::unordered_map<std::string, GameType::Entity>& entities,
46                                           std::unordered_map<std::string, GameType::Structure>& structures,
47                                           std::unordered_map<std::string, GameType::FloorType>& floors);
48     bool _isReachable(Coordinate position) const;
49     void _getTargets(Coordinate center, unsigned int range, std::vector<Coordinate>& targets,
50                     bool detectUnreachableTargets) const;
51 public:
52     explicit Map(MapFileReader& mapFile, std::list<Coordinate>& priests);
53
54     Map() = default; /*Crea mapa de 0x0, lo usamos para las pruebas*/
55
56     //Ataca la tile y retorna el resultado del ataque, el booleano indica si el ataque
57     //fue realizado (true) o no (false)

```

jul 21, 20 15:47	Map.h	Page 2/3
58	<code>std::pair<AttackResult, bool> attackTile(int damage, unsigned int level, bool isAPlayer,</code>	
59	<code>Coordinate coordinate);</code>	
60		
61	<code>//Almacena en el vector la cantidad de targets de un monstruo en un cuadrado centrado en</code>	
62	<code>//center de lado 2*range+1</code>	
63	<code>void getMoveTargets(Coordinate center, unsigned int range, std::vector<Coordinate>& targets) const;</code>	
64		
65	<code>//Almacena en el vector la cantidad de targets de un monstruo para atacar en un cuadrado centrado en</code>	
66	<code>//center de lado 2*range+1</code>	
67	<code>void getAttackTargets(Coordinate center, unsigned int range, std::vector<Coordinate>& targets) const;</code>	
68		
69	<code>//Almacena en la lista el camino que se debe seguir para llegar a la coordenada deseada</code>	
70	<code>//Si existe un camino retorna true y la informacion es guardada en path, sino retorna</code>	
71	<code>//false y no guarda nada</code>	
72	<code>bool getPath(Coordinate currentPosition, Coordinate desiredPosition, std::list<Coordinate>& path) const;</code>	
73		
74	<code>//Intenta agregar la entity al tile que se encuentra en la coordenada recibida apropiandose</code>	
75	<code>//del shared_ptr, si la coordenada es invalida tira invalid_argument y no se apropia del puntero</code>	
76	<code>//Si la posicion es ocupable entonces se apropia del puntero, sino tira TPEception</code>	
77	<code>void addEntity(Coordinate position, std::shared_ptr<Entity>^ entity);</code>	
78		
79	<code>//Toma el primer item almacenado en el tile que se encuentra en la coordenada</code>	
80	<code>//pasada, lo elimina de la tile y lo retorna, si la coordenada es invalida se</code>	
81	<code>//tira una exception de invalid_argument</code>	
82	<code>//Si el tile no tiene items retorna un shared_ptr que almacena nullptr</code>	
83	<code>std::shared_ptr<Item> removeItem(Coordinate position);</code>	
84		
85	<code>//Elimina la entity almacenada en la coordenada, liberando sus recursos y permitiendo</code>	
86	<code>//el almacenamiento de otra entity, si la coordenada es invalida tira invalid_argument</code>	
87	<code>void removeEntity(Coordinate position);</code>	
88		
89	<code>//Intenta mover la entity de starting a final position, si finalPosition esta ocupada</code>	
90	<code>//entonces retorna false, sino retorna true</code>	
91	<code>//Si alguna de las coordenadas es invalida tira invalid_argument</code>	
92	<code>void moveEntity(Coordinate startingPosition, Coordinate finalPosition);</code>	
93		
94	<code>//Retorna true si el lugar puede ser ocupado por una entity, false en caso contrario</code>	
95	<code>bool isPlaceAvailable(Coordinate position) const;</code>	
96		
97	<code>//Agrega los items de la lista al tile al tile que se encuentra en la coordenada recibida apropiandose de la lista,</code>	
98	<code>//si la coordenada es invalida tira invalid_argument y no se apropia del puntero</code>	
99	<code>//Se pueden guardar smart_pointers que contengan nullptr, pero no se deberia hacer, el chequeo</code>	
100	<code>//de si se esta guardando null o no tiene que venir de afuera</code>	
101	<code>void addItemToTile(std::list<std::shared_ptr<Item>>^ items, Coordinate position);</code>	
102		

jul 21, 20 15:47	Map.h	Page 3/3
103	<code>//Agrega el item al tile que se encuentra en la coordenada recibida apropiandose del shared_ptr,</code>	
104	<code>//si la coordenada es invalida tira invalid_argument y no se apropia del puntero</code>	
105	<code>//Se pueden guardar smart_pointers que contengan nullptr, pero no se deberia hacer, el chequeo</code>	
106	<code>//de si se esta guardando null o no tiene que venir de afuera</code>	
107	<code>void addItemToTile(std::shared_ptr<Item>^ item, Coordinate position);</code>	
108		
109	<code>//Retorna una coordenada aleatoria en la que puede ponerse un monstruo</code>	
110	<code>Coordinate getMonsterCoordinate();</code>	
111		
112	<code>//Delega el comportamiento a la entity que guarda, si es que guarda una</code>	
113	<code>//unsigned int list(Player& player, std::list<ProductData>& products, Coordinate coordinate);</code>	
114	<code>void list(Player& player, Coordinate coordinate);</code>	
115		
116	<code>//Delega el comportamiento a la entity que guarda, si es que guarda una</code>	
117	<code>void withdraw(Player& player, const std::string& itemName, Coordinate coordinate);</code>	
118		
119	<code>//Delega el comportamiento a la entity que guarda, si es que guarda una</code>	
120	<code>void deposit(Player& player, const std::string& itemName, Coordinate coordinate);</code>	
121		
122	<code>//Delega el comportamiento a la entity que guarda, si es que guarda una</code>	
123	<code>void buy(Player& player, const std::string& itemName, Coordinate coordinate);</code>	
124		
125	<code>//Delega el comportamiento a la entity que guarda, si es que guarda una</code>	
126	<code>void sell(Player& player, const std::string& itemName, Coordinate coordinate);</code>	
127		
128	<code>//Guarda el estado actual del mapa para que pueda ser enviado a un nuevo cliente</code>	
129	<code>void operator>>(std::stringstream& mapBuffer) const;</code>	
130		
131	<code>//Retorna un pair que almacena el tipo del item y su id, si la coordenada es inexistente</code>	
132	<code>//guarda -2 en el id (second), si el tile no tiene items guarda -1, sino guarda el id del item</code>	
133	<code>//std::pair<GameType::ItemType, int32_t> peekShowedItemData(Coordinate coordinate);</code>	
134	<code>const Item* peekShowedItemData(Coordinate coordinate);</code>	
135		
136	<code>//Retorna una posicion disponible alrededor de la posicion recibida</code>	
137	<code>Coordinate getSpawnCoordinateArroundPosition(Coordinate reference);</code>	
138		
139	<code>void requestRestore(Player& player, Coordinate target);</code>	
140		
141	<code>//Retorna una coordenada aleatoria alrededor de reference que no este ocupada ni</code>	
142	<code>//pertenezca a una ciudad</code>	
143	<code>Coordinate getMonsterRandomPosition(Coordinate reference) const;</code>	
144		
145	<code>virtual ~Map() = default; /*Para que FakeIt lo pueda mockear*/</code>	
146	<code>};</code>	
147		
148		
149	<code>#endif //ARGENTUM_MAP_H</code>	

jul 21, 20 15:47

Map.cpp

Page 1/7

```

1  //
2  // Created by agustin on 7/6/20.
3  //
4
5  #include <queue>
6  #include <unordered_map>
7  #include <memory>
8  #include "Map.h"
9  #include "../Entities/AttackResult.h"
10 #include "../Config/Calculator.h"
11 #include "../Entities/Citizens/CitizenFactory.h"
12 #include "../Config/MapFileReader.h"
13 #include <msgpack.hpp>
14
15 #define RESPAWN_RANGE 3
16
17 ///////////////////////////////////PRIVATE////////////////////////////////////
18
19 //Retorna la distancia (siempre positiva) entre las dos coordenadas
20 unsigned int Map::_getDistance(Coordinate a, Coordinate b) {
21     return std::abs((a.iPosition - b.iPosition) + (a.jPosition - b.jPosition));
22 }
23
24
25 //Indica si la coordenada esta en el rango de posiciones del mapa
26 bool Map::_isCoordinateValid(Coordinate coordinate) const {
27     return (coordinate.jPosition ≥ 0) ^ (coordinate.jPosition < (int)tiles[0].size())
28           ^ (coordinate.iPosition ≥ 0) ^ (coordinate.iPosition < (int)tiles.size());
29 }
30
31
32 bool Map::_areCoordinatesEqual(Coordinate a, Coordinate b){
33     return (a.iPosition == b.iPosition) ^ (a.jPosition == b.jPosition);
34 }
35
36
37 void Map::_buildSearchRegion(Coordinate center, unsigned int range, Coordinate&
38 topLeft, Coordinate& bottomRight) const {
39     Coordinate aux{};
40     aux.iPosition = static_cast<int>(center.iPosition - range);
41     aux.jPosition = static_cast<int>(center.jPosition - range);
42     topLeft = _getValidCoordinate(aux);
43     aux.iPosition = static_cast<int>(center.iPosition + range);
44     aux.jPosition = static_cast<int>(center.jPosition + range);
45     bottomRight = _getValidCoordinate(aux);
46 }
47
48 Coordinate Map::_getValidCoordinate(Coordinate coordinate) const {
49     if (coordinate.jPosition ≥ (int)tiles[0].size()) {
50         coordinate.jPosition = (int)tiles[0].size() - 1;
51     } else if (coordinate.jPosition < 0) {
52         coordinate.jPosition = 0;
53     }
54     if (coordinate.iPosition ≥ (int)tiles.size()) {
55         coordinate.iPosition = (int)tiles.size() - 1;
56     } else if (coordinate.iPosition < 0) {
57         coordinate.iPosition = 0;
58     }
59     return coordinate;
60 }
61
62 //Guarda en nodes y parentsAndChilds los nodos correspondientes, revisando los
63 //nodos que se encuentren adyacentes a referencia, tambien actualiza las distanc

```

jul 21, 20 15:47

Map.cpp

Page 2/7

```

64 //de los nodos de ser necesario
65 void Map::_storeAdjacentPositions(
66     PointAndDistance refference, std::unordered_map<Coordinate, unsigned int>
67     & distances,
68     std::unordered_map<Coordinate, Coordinate>& parentsAndChilds,
69     std::priority_queue<PointAndDistance, std::vector<PointAndDistance>,
70     InverseCoordinateDistance>& nodes,
71     Coordinate destination) const {
72     Coordinate topLeft{}, bottomRight{};
73     PointAndDistance aux{};
74     _buildSearchRegion(refference.point, 1, topLeft, bottomRight);
75     for (int i = topLeft.iPosition; i ≤ bottomRight.iPosition; ++i) {
76         for (int j = topLeft.jPosition; j ≤ bottomRight.jPosition; ++j) {
77             aux.point.iPosition = i;
78             aux.point.jPosition = j;
79             aux.distance = _getDistance(refference.point, aux.point);
80             //if (esta a distancia 1 y (la posicion es alcanzable o tiene un jug
81             ador) y no esta en una ciudad)
82             if ((aux.distance == 1) ^ (tiles[i][j].isAvailable() ^
83                 tiles[i][j].hasMonsterTarget()) ^ !tiles[i][j].isInCity()) {
84                 aux.distance += refference.distance + _getDistance(aux.point, de
85                 stination);
86                 if ((distances.count(aux.point) == 0) ^
87                     (distances.at(aux.point) > aux.distance)) {
88                     nodes.push(aux);
89                     distances[aux.point] = aux.distance;
90                     parentsAndChilds[aux.point] = refference.point;
91                 }
92             }
93         }
94     }
95
96 void Map::_storePath(Coordinate initialPosition, Coordinate desiredPosition,
97     const std::unordered_map<Coordinate, Coordinate>& parentsAnd
98     Childs,
99     std::list<Coordinate>& path) {
100     Coordinate aux = desiredPosition;
101     while (!_areCoordinatesEqual(aux, initialPosition)) {
102         path.push_front(aux);
103         aux = parentsAndChilds.at(aux);
104     }
105
106 bool Map::_isReachable(Coordinate position) const {
107     Coordinate topLeft{};
108     Coordinate bottomRight{};
109     Coordinate aux{};
110     _buildSearchRegion(position, 1, topLeft, bottomRight);
111     for (int i = topLeft.iPosition; i ≤ bottomRight.iPosition; ++i) {
112         for (int j = topLeft.jPosition; j ≤ bottomRight.jPosition; ++j) {
113             aux = {i, j};
114             if ((_getDistance(position, aux) == 1) ^ (tiles[i][j].isAvailable() ^
115                 !tiles[i][j].isInCity())) {
116                 return true;
117             }
118         }
119     }
120     return false;
121 }
122
123 void Map::_initializeConstructorMaps(

```

jul 21, 20 15:47

Map.cpp

Page 3/7

```

124     std::unordered_map<std::string, GameType::Entity> &entities,
125     std::unordered_map<std::string, GameType::Structure> &structures,
126     std::unordered_map<std::string, GameType::FloorType> &floors) {
127     entities = {"Nothing", GameType::Entity::GUARD}, {"Priest", GameType::Entity::P
RIEST},
128             {"Trader", GameType::Entity::TRADER}, {"Banker", GameType::Entity::BA
NKER}};
129     structures = {"BoneGuy", GameType::Structure::BONE_GUY}, {"BrokenRipStone", Ga
meType::Structure::BROKEN_RIP_STONE},
130             {"Bush", GameType::Structure::BUSH}, {"DeadBush", GameType::Structure
::DEAD_BUSH},
131             {"DeadGuy", GameType::Structure::DEAD_GUY}, {"DeadTree", GameType::Str
ucture::DEAD_TREE},
132             {"FatTree", GameType::Structure::FAT_TREE}, {"HangedGuy", GameType::St
ructure::HANGED_GUY},
133             {"House1", GameType::Structure::HOUSE1}, {"House2", GameType::Structu
re::HOUSE2}, {"House3", GameType::Structure::HOUSE3}, {"LongTree", GameType::Struct
ure::LONG_TREE},
134             {"PalmTree", GameType::Structure::PALM_TREE}, {"RipStone", GameType::St
ructure::RIP_STONE},
135             {"Tree", GameType::Structure::TREE}, {"VeryDeadGuy", GameType::Structu
re::VERY_DEAD_GUY},
136             {"SunkenColumn", GameType::Structure::SUNKEN_COLUMN}, {"SunkenShip", Ga
meType::Structure::SUNKEN_SHIP},
137             {"Nothing", GameType::Structure::NO_STRUCTURE}};
138
139     floors = {"Grass0", GameType::FloorType::GRASS0}, {"Grass1", GameType::FloorT
ype::GRASS1},
140             {"Grass2", GameType::FloorType::GRASS2}, {"Grass3", GameType::FloorTyp
e::GRASS3},
141             {"Sand", GameType::FloorType::SAND}, {"Water0", GameType::FloorType::
WATER0},
142             {"Water1", GameType::FloorType::WATER1}, {"Water2", GameType::FloorTyp
e::WATER2},
143             {"Water3", GameType::FloorType::WATER3}, {"PrettyRoad0", GameType::Floo
rType::PRETTY_ROAD0},
144             {"PrettyRoad1", GameType::FloorType::PRETTY_ROAD1}, {"PrettyRoad2", GameT
ype::FloorType::PRETTY_ROAD2},
145             {"PrettyRoad3", GameType::FloorType::PRETTY_ROAD3}, {"PrettyGrass0", Gam
eType::FloorType::PRETTY_GRASS0},
146             {"PrettyGrass1", GameType::FloorType::PRETTY_GRASS1}, {"PrettyGrass2", Gam
eType::FloorType::PRETTY_GRASS2},
147             {"PrettyGrass3", GameType::FloorType::PRETTY_GRASS3}, {"DeadGrass0", Gam
eType::FloorType::DEAD_GRASS0},
148             {"DeadGrass1", GameType::FloorType::DEAD_GRASS1}, {"DeadGrass2", GameTy
pe::FloorType::DEAD_GRASS2},
149             {"DeadGrass3", GameType::FloorType::DEAD_GRASS3}, {"DarkWater0", GameTy
pe::FloorType::DARK_WATER0},
150             {"DarkWater1", GameType::FloorType::DARK_WATER1}, {"DarkWater2", GameTy
pe::FloorType::DARK_WATER2},
151             {"DarkWater3", GameType::FloorType::DARK_WATER3}};
152 }
153
154 void Map::_getTargets(Coordinate center, unsigned int range, std::vector<Coordin
ate> &targets,
155                     bool detectUnreachableTargets) const {
156     Coordinate topLeft{}, bottomRight{}, aux{};
157     _buildSearchRegion(center, range, topLeft, bottomRight);
158     for (int i = topLeft.iPosition; i ≤ bottomRight.iPosition; ++i) {
159         for (int j = topLeft.jPosition; j ≤ bottomRight.jPosition; ++j) {
160             if (tiles[i][j].hasMonsterTarget() ^ !tiles[i][j].isInCity() ^
161                 (_isReachable({i, j}) v detectUnreachableTargets)) {
162                 aux.iPosition = i;
163                 aux.jPosition = j;
164                 targets.push_back(aux);
165             }

```

jul 21, 20 15:47

Map.cpp

Page 4/7

```

166     }
167 }
168 }
169 }
170
171
172
173 //////////////////////////////////////////////////PUBLIC////////////////////////////////////
174
175 Map::Map(MapFileReader &mapFile, std::list<Coordinate> & priests) {
176     CitizenFactory citizenFactory;
177     MapSize mapSize = mapFile.getMapDimensions();
178     TileInfo aux{};
179     std::shared_ptr<Entity> citizen;
180     std::unordered_map<std::string, GameType::Entity> entities;
181     std::unordered_map<std::string, GameType::Structure> structures;
182     std::unordered_map<std::string, GameType::FloorType> floors;
183     _initializeConstructorMaps(entities, structures, floors);
184     //tiles.resize(mapSize.height, std::vector<Tile>(mapSize.width));
185     GameType::Entity auxEntity{};
186     for (unsigned int i = 0; i < mapSize.height; ++i) {
187         tiles.emplace_back();
188         for (unsigned int j = 0; j < mapSize.width; ++j) {
189             aux = mapFile.getTileInfo(i, j);
190             if (aux.entityType == "Nothing") {
191                 citizen.reset();
192             } else {
193                 auxEntity = entities.at(aux.entityType);
194                 citizenFactory.storeCitizen(citizen, auxEntity,
195                     {static_cast<int>(i), static_cast<int>(j)});
196                 if (auxEntity == GameType::PRIEST) {
197                     priests.push_back({static_cast<int>(i), static_cast<int>(j)})
198                 }
199             }
200             tiles.at(i).emplace_back(aux.isOccupable, aux.isFromCity, floors.at(
aux.tileType), structures
201                 .at(aux.structureType), std::move(citizen));
202         }
203     }
204 }
205
206
207 std::pair<AttackResult, bool> Map::attackTile(int damage, unsigned int level, bo
ol isAPlayer,
208                                             Coordinate coordinate) {
209     return tiles[coordinate.iPosition][coordinate.jPosition].attacked(damage, le
vel, isAPlayer);
210 }
211
212 void Map::getMoveTargets(Coordinate center, unsigned int range, std::vector<Coor
dinate> & targets) const {
213     _getTargets(center, range, targets, false);
214 }
215
216 void Map::getAttackTargets(Coordinate center, unsigned int range,
217                           std::vector<Coordinate> & targets) const {
218     _getTargets(center, range, targets, true);
219 }
220
221 bool Map::getPath(Coordinate currentPosition, Coordinate desiredPosition, std::l
ist<Coordinate> & path) const {
222     std::priority_queue<PointAndDistance, std::vector<PointAndDistance>, Inverse
CoordinateDistance> nodes;
223
224

```

jul 21, 20 15:47

Map.cpp

Page 5/7

```

225 //Key: hijo, Dato: padre
226 std::unordered_map<Coordinate, Coordinate> parentsAndChilds;
227
228 //Key: Posicion, Dato: distancia
229 std::unordered_map<Coordinate, unsigned int> distances;
230
231 PointAndDistance aux{};
232 aux.point = currentPosition;
233 aux.distance = 0;
234 nodes.push(aux);
235 while (!nodes.empty()) {
236     aux = nodes.top();
237     nodes.pop();
238     if (_areCoordinatesEqual(aux.point, desiredPosition)) {
239         _storePath(currentPosition, desiredPosition, parentsAndChilds, path)
240     };
241     return true;
242 }
243 _storeAdjacentPositions(aux, distances, parentsAndChilds, nodes, desired
Position);
244 }
245 return false;
246 }
247 void Map::addEntity(Coordinate position, std::shared_ptr<Entity> ^entity) {
248     if (!_isCoordinateValid(position)) {
249         throw (std::invalid_argument("Out of bounds coordinate"));
250     }
251     entity->setPosition(position);
252     tiles[position.iPosition][position.jPosition].addEntity(std::move(entity));
253 }
254
255 std::shared_ptr<Item> Map::removeItem(Coordinate position) {
256     if (!_isCoordinateValid(position)) {
257         throw (std::invalid_argument("Out of bounds coordinate"));
258     }
259     return tiles[position.iPosition][position.jPosition].removeItem();
260 }
261
262 void Map::removeEntity(Coordinate position) {
263     if (!_isCoordinateValid(position)) {
264         throw (std::invalid_argument("Out of bounds coordinate"));
265     }
266     tiles[position.iPosition][position.jPosition].removeEntity();
267 }
268
269 void Map::moveEntity(Coordinate startingPosition, Coordinate finalPosition) {
270     if ((!_isCoordinateValid(startingPosition)) &
(!_isCoordinateValid(finalPosition))) {
271         return;
272     }
273     if (!tiles[finalPosition.iPosition][finalPosition.jPosition].isAvailable())
274     {
275         return;
276     }
277     Tile& tile = tiles[finalPosition.iPosition][finalPosition.jPosition];
278     tile.moveEntity(tiles[startingPosition.iPosition][startingPosition.jPosition
],
279                 finalPosition);
280 }
281
282 bool Map::isPlaceAvailable(Coordinate position) const {
283     return _isCoordinateValid(position) &
284         tiles[position.iPosition][position.jPosition].isAvailable();
285 }
286

```

jul 21, 20 15:47

Map.cpp

Page 6/7

```

287 void Map::addItemToTile(std::list<std::shared_ptr<Item>>^ items, Coordinate po
sition) {
288     if (!_isCoordinateValid(position)) {
289         throw (std::invalid_argument("Out of bounds coordinate"));
290     }
291     tiles[position.iPosition][position.jPosition].addItem(std::move(items));
292 }
293
294 void Map::addItemToTile(std::shared_ptr<Item> ^item, Coordinate position) {
295     if (!_isCoordinateValid(position)) {
296         throw (std::invalid_argument("Out of bounds coordinate"));
297     }
298     tiles[position.iPosition][position.jPosition].addItem(std::move(item));
299 }
300
301
302 Coordinate Map::getMonsterCoordinate() {
303     unsigned int xPosition = Calculator::getRandomInt(0, (int)(tiles.size() - 1)
);
304     unsigned int yPosition = Calculator::getRandomInt(0, (int)(tiles[0].size() -
1));
305     while ((!tiles[xPosition][yPosition].isAvailable()) & (tiles[xPosition][yPo
sition].isInCity())) {
306         xPosition = Calculator::getRandomInt(0, (int)(tiles.size() - 1));
307         yPosition = Calculator::getRandomInt(0, (int)(tiles[0].size() - 1));
308     }
309     return {static_cast<int>(xPosition), static_cast<int>(yPosition)};
310 }
311
312 void Map::list(Player &player, Coordinate coordinate) {
313     if (!_isCoordinateValid(coordinate)) {
314         tiles[coordinate.iPosition][coordinate.jPosition].list(player);
315     }
316 }
317
318 void Map::withdraw(Player &player, const std::string &itemName, Coordinate coord
inate) {
319     if (!_isCoordinateValid(coordinate)) {
320         tiles[coordinate.iPosition][coordinate.jPosition].withdraw(player, itemN
ame);
321     }
322 }
323
324 void Map::deposit(Player &player, const std::string &itemName, Coordinate coordi
nate) {
325     if (!_isCoordinateValid(coordinate)) {
326         tiles[coordinate.iPosition][coordinate.jPosition].deposit(player, itemNa
me);
327     }
328 }
329
330 void Map::buy(Player &player, const std::string &itemName, Coordinate coordinate
) {
331     if (!_isCoordinateValid(coordinate)) {
332         tiles[coordinate.iPosition][coordinate.jPosition].buy(player, itemName);
333     }
334 }
335
336 void Map::sell(Player &player, const std::string &itemName, Coordinate coordinat
e) {
337     if (!_isCoordinateValid(coordinate)) {
338         tiles[coordinate.iPosition][coordinate.jPosition].sell(player, itemName)
;
339     }
340 }
341

```


jul 21, 20 15:47

Map.cpp

Page 7/7

```

342 void Map::operator>>(std::stringstream &mapBuffer) const {
343     msgpack::type::tuple<int32_t, int32_t> tileSize(tiles.size(), tiles[0].size(
    ));
344     msgpack::pack(mapBuffer, tileSize);
345     for (const auto & row : tiles) {
346         for (const auto & tile : row) {
347             tile >> mapBuffer;
348         }
349     }
350 }
351
352 const Item* Map::peekShowedItemData(Coordinate coordinate) {
353     if (!_isCoordinateValid(coordinate)) {
354         throw std::invalid_argument("Invalid coordinate in peekShoedItemData");
355     }
356     return tiles[coordinate.iPosition][coordinate.jPosition].peekShowedItemData(
    );
357 }
358
359 Coordinate Map::getSpawnCoordinateArroundPosition(Coordinate reference) {
360     Coordinate topLeft{}, bottomRight{};
361     _buildSearchRegion(reference, RESPAWN_RANGE, topLeft, bottomRight);
362     for (int i = topLeft.iPosition; i ≤ bottomRight.iPosition; ++i) {
363         for (int j = topLeft.jPosition; j ≤ bottomRight.jPosition; ++j) {
364             if (tiles[i][j].isAvailable() ^ tiles[i][j].isInCity()) {
365                 return {i, j};
366             }
367         }
368     }
369     return {-1, -1};
370 }
371
372 Coordinate Map::getMonsterRandomPosition(Coordinate reference) const {
373     std::vector<Coordinate> positions;
374     Coordinate topLeft{}, bottomRight{}, aux{};
375     _buildSearchRegion(reference, 1, topLeft, bottomRight);
376     for (int i = topLeft.iPosition; i ≤ bottomRight.iPosition; ++i) {
377         for (int j = topLeft.jPosition; j ≤ bottomRight.jPosition; ++j) {
378             aux = {i, j};
379             if ((reference.calculateDistance(aux) == 1) ^ (tiles[i][j].isAvailab
le() ^
380                 (¬tiles[i][j].isInCity()))) {
381                 positions.push_back(aux);
382             }
383         }
384     }
385     if (positions.empty()) {
386         return {-1, -1};
387     }
388     return positions[Calulator::getRandomInt(0, positions.size() - 1)];
389 }
390
391 void Map::requestRestore(Player &player, Coordinate target) {
392     if (!_isCoordinateValid(target)) {
393         tiles[target.iPosition][target.jPosition].requestRestore(player);
394     }
395 }
396
397

```

jul 21, 20 15:47

InverseCoordinateDistance.h

Page 1/1

```

1 //
2 // Created by agustin on 10/6/20.
3 //
4
5 #ifndef ARGENTUM_INVERSECOORDINATEDISTANCE_H
6 #define ARGENTUM_INVERSECOORDINATEDISTANCE_H
7
8
9 #include "PointAndDistance.h"
10
11 //Functor utilizado para obtener el nodo de menor distancia en la priority_queue
12 //para el pathfinding
13 class InverseCoordinateDistance {
14 public:
15     bool operator()(const PointAndDistance& a, const PointAndDistance& b);
16 };
17
18
19 #endif //ARGENTUM_INVERSECOORDINATEDISTANCE_H

```

jul 21, 20 15:47

InverseCoordinateDistance.cpp

Page 1/1

```

1 //
2 // Created by agustin on 10/6/20.
3 //
4
5 #include "InverseCoordinateDistance.h"
6
7 bool InverseCoordinateDistance::operator()(const PointAndDistance &a,
8                                           const PointAndDistance &b) {
9     return a.distance > b.distance;
10 }

```

jul 21, 20 15:47

Coordinate.h

Page 1/1

```

1 //
2 // Created by agustin on 6/6/20.
3 //
4
5 #ifndef ARGENTUM_COORDINATE_H
6 #define ARGENTUM_COORDINATE_H
7
8 #include <functional>
9 #include <cstdlib>
10
11 //Struct auxiliar para facilitar el paso de coordenadas, debido a esto no
12 //tiene ningun comportamiento y sus atributos son publicos
13 struct Coordinate {
14     int iPosition;
15     int jPosition;
16
17     bool operator==(const Coordinate& other) {
18         return ((iPosition == other.iPosition) ^ (jPosition == other.jPosition));
19     }
20
21     bool operator!=(const Coordinate& other) {
22         return !((iPosition == other.iPosition) ^ (jPosition == other.jPosition))
23         ;
24     }
25
26     unsigned int calculateDistance(const Coordinate& other) {
27         return std::abs(iPosition - other.iPosition) + std::abs(jPosition - othe
28         r.jPosition);
29     };
30
31 namespace std {
32     template <> struct hash<Coordinate> {
33         size_t operator()(const Coordinate& x) const {
34             return (x.jPosition << 15) + x.jPosition;
35         }
36     };
37
38     template <> struct equal_to<Coordinate> {
39         size_t operator()(const Coordinate& x, const Coordinate& y) const {
40             return (x.iPosition == y.iPosition) ^ (x.jPosition == y.jPosition);
41         }
42     };
43
44 #endif //ARGENTUM_COORDINATE_H

```

jul 21, 20 15:47

UseReturnData.h

Page 1/1

```

1 //
2 // Created by agustin on 3/7/20.
3 //
4
5 #ifndef ARGENTUM_USERRETURNDATA_H
6 #define ARGENTUM_USERRETURNDATA_H
7
8 #include "Item.h"
9
10 /*Este struct es util para cuando mandamos los updates a los players de lo que
11 * se equipo otro jugador*/
12
13 struct UseReturnData {
14     GameType::EquipmentPlace equipmentPlace;
15     int32_t id;
16 };
17
18
19 #endif //ARGENTUM_USERRETURNDATA_H

```

jul 21, 20 15:47

Potion.h

Page 1/1

```

1 //
2 // Created by agustin on 19/6/20.
3 //
4
5 #ifndef ARGENTUM_POTION_H
6 #define ARGENTUM_POTION_H
7
8
9 #include "../Item.h"
10 #include "../Config/Configuration.h"
11
12 class ItemTests;
13
14 /*Esta clase encapsula el comportamiento global de las pociones que
15 * son consumibles por el player (pociones de vida/mana)*/
16
17 class Potion: public Item {
18 protected:
19     unsigned int recoveryValue;
20
21     friend ItemTests;
22
23 public:
24     explicit Potion(GameType::Potion potion);
25
26     /*Indica que una vez que son usadas las pociones son descartadas*/
27     GameType::EquipmentPlace use(Player& player) override;
28
29     /*Debe llamar a la funcion de player que restaura el atributo correspondient
30 e*/
31     virtual void restoreStat(Player& player) = 0;
32
33     virtual ~Potion() = default;
34 };
35
36 #endif //ARGENTUM_POTION_H

```

jul 21, 20 15:47

Potion.cpp

Page 1/1

```

1  //
2  // Created by agustin on 19/6/20.
3  //
4
5  #include "Potion.h"
6
7
8  GameType::EquipmentPlace Potion::use(Player &player) {
9      restoreStat(player);
10     return GameType::EQUIPMENT_PLACE_NONE;
11 }
12
13 Potion::Potion(GameType::Potion potion):
14     Item(GameType::ITEM_TYPE_POTION, Configuration::getInstance().co
15 nfigPotionData(potion).name) {
16     recoveryValue = Configuration::getInstance().configPotionData(potion).recove
17 ryValue;
18     id = potion;
19 }

```

jul 21, 20 15:47

ManaPotion.h

Page 1/1

```

1  //
2  // Created by agustin on 19/6/20.
3  //
4
5  #ifndef ARGENTUM_MANAPOTION_H
6  #define ARGENTUM_MANAPOTION_H
7
8
9  #include "Potion.h"
10
11 /*Esta clase representa la pocion de mana que puede consumir el player*/
12
13 class ManaPotion: public Potion {
14 private:
15     void restoreStat(Player& player) override;
16
17 public:
18     explicit ManaPotion(): Potion(GameType::MANA_POTION) {}
19     ~ManaPotion() override;
20 };
21
22
23 #endif //ARGENTUM_MANAPOTION_H

```

jul 21, 20 15:47

ManaPotion.cpp

Page 1/1

```

1  //
2  // Created by agustin on 19/6/20.
3  //
4
5  #include "ManaPotion.h"
6  #include "../Entities/Player.h"
7
8  void ManaPotion::restoreStat(Player &player) {
9      player.restoreMana(recoveryValue);
10 }
11
12 ManaPotion::~ManaPotion() = default;

```

jul 21, 20 15:47

HealthPotion.h

Page 1/1

```

1  //
2  // Created by agustin on 19/6/20.
3  //
4
5  #ifndef ARGENTUM_HEALTHPOTION_H
6  #define ARGENTUM_HEALTHPOTION_H
7
8
9  #include "Potion.h"
10
11 /*Esta clase representa la pocion de vida que puede consumir el player*/
12
13 class HealthPotion: public Potion {
14 private:
15     void restoreStat(Player& player) override;
16 public:
17     explicit HealthPotion(): Potion(GameType::HEALTH_POTION) {}
18     ~HealthPotion() override;
19 };
20
21
22 #endif //ARGENTUM_HEALTHPOTION_H

```

jul 21, 20 15:47

HealthPotion.cpp

Page 1/1

```

1  //
2  // Created by agustin on 19/6/20.
3  //
4
5  #include "HealthPotion.h"
6  #include "../Entities/Player.h"
7
8  void HealthPotion::restoreStat(Player &player) {
9      player.restoreLife(recoveryValue);
10 }
11
12 HealthPotion::~HealthPotion() = default;

```

jul 21, 20 15:47

Gold.h

Page 1/1

```

1  //
2  // Created by agustin on 6/6/20.
3  //
4
5  #ifndef ARGENTUM_GOLD_H
6  #define ARGENTUM_GOLD_H
7
8
9  #include "../Item.h"
10
11 class ItemTests;
12
13 /*Clase que representa un puniado de oro en el piso*/
14
15 class Gold: public Item {
16 private:
17     unsigned int amount;
18
19     friend ItemTests;
20
21 public:
22     explicit Gold(unsigned int amount);
23
24     /*Retorna la posicion donde se equipa (como es oro no se equipa en ningun la
25 do sino que
26 * en realidad le sumamos a la cantidad de oro que guarda el player)*/
27     GameType::EquipmentPlace use(Player& player) override;
28
29     /*Retorna la cantidad de oro que representa la instancia Gold*/
30     unsigned int getAmount() const;
31
32     /* Retorna true, no es la solucion mas limpia pero era la mas sencilla de
33 ado en si
34 * implementar. Esto soluciona el tema donde el gold en realidad no va equip
35 * asi que no lo agregamos al inventario, pero como hereda de item necesitab
36 amos
37 * poder distinguir si era o no oro para sumarle a la variable del player*/
38     bool isGold() const override;
39 };
40
41 #endif //ARGENTUM_GOLD_H

```

jul 21, 20 15:47

Gold.cpp

Page 1/1

```

1 //
2 // Created by agustin on 6/6/20.
3 //
4
5 #include "Gold.h"
6 #include "../Config/Configuration.h"
7
8 Gold::Gold(unsigned int amount): Item(GameType::ITEM_TYPE_GOLD,
9                                     Configuration::getInstance().configGetGoldName()/*, 0*/){
10     this->amount = amount;
11 }
12
13 GameType::EquipmentPlace Gold::use(Player &player) {
14     return GameType::EQUIPMENT_PLACE_NONE;
15 }
16
17 unsigned int Gold::getAmount() const {
18     return amount;
19 }
20
21 bool Gold::isGold() const {
22     return true;
23 }

```

jul 21, 20 15:47

ItemsFactory.h

Page 1/2

```

1 //
2 // Created by agustin on 16/6/20.
3 //
4
5 #ifndef ARGENTUM_ITEMSFACTORY_H
6 #define ARGENTUM_ITEMSFACTORY_H
7
8 #include <unordered_map>
9 #include <memory>
10 #include <vector>
11 #include "../libs/GameEnums.h"
12
13 class Item;
14 class ItemTests;
15
16 /*Esta es una factory de items, para no estar hardcodeando cada vez que necesite
17 mos
18 * crear un item particular (ya que como se crea un item depende del tipo)*/
19
20 typedef void (*objectCreator) (std::shared_ptr<Item>&);
21
22 class ItemsFactory {
23 private:
24     std::unordered_map<std::string, objectCreator> itemsCreators;
25     std::vector<const std::string*> itemsNames;
26
27     friend ItemTests;
28
29 private:
30     ItemsFactory();
31
32     static void _storeBlueTunic(std::shared_ptr<Item>& item);
33     static void _storeLeatherArmor(std::shared_ptr<Item>& item);
34     static void _storePlateArmor(std::shared_ptr<Item>& item);
35     static void _storeKingArmor(std::shared_ptr<Item> &item);
36
37     static void _storeHood(std::shared_ptr<Item>& item);
38     static void _storeIronHelmet(std::shared_ptr<Item>& item);
39     static void _storeMagicHat(std::shared_ptr<Item>& item);
40
41     static void _storeIronShield(std::shared_ptr<Item>& item);
42     static void _storeTurtleShield(std::shared_ptr<Item>& item);
43
44     static void _storeAshRod(std::shared_ptr<Item>& item);
45     static void _storeCompositeBow(std::shared_ptr<Item>& item);
46     static void _storeElvenFlute(std::shared_ptr<Item>& item);
47     static void _storeGnarledStaff(std::shared_ptr<Item>& item);
48     static void _storeLinkedStaff(std::shared_ptr<Item>& item);
49     static void _storeLongsword(std::shared_ptr<Item>& item);
50     static void _storeSimpleBow(std::shared_ptr<Item>& item);
51     static void _storeWarhammer(std::shared_ptr<Item>& item);
52     static void _storeAxe(std::shared_ptr<Item>& item);
53
54     static void _storeManaPotion(std::shared_ptr<Item>& item);
55     static void _storeHealthPotion(std::shared_ptr<Item>& item);
56
57     static void _storeRandomPotion(std::shared_ptr<Item>& item);
58
59     static void _storeGold(std::shared_ptr<Item>& item, unsigned int amount);
60
61 public:
62     static ItemsFactory& getInstance();
63
64     /*Guarda una instancia del item pedido en item, si el nombre del item pasado
65     no existe entonces tira la excepcion out_of_range*/

```

jul 21, 20 15:47

ItemsFactory.h

Page 2/2

```
66     void storeItemInstance(const std::string& itemName, std::shared_ptr<Item>& i
tem);
67
68     /*Guarda una instancia del item pedido en item, si el nombre del item pasado
69     no existe entonces tira la excepcion out_of_range*/
70     void storeItemInstance(GameType::ItemType type, int32_t instance,
71                             std::shared_ptr<Item> &item);
72
73     /*Almacena un item aleatorio en item, goldMultiplier es el valor por el que
74     se
75     multiplica el porcentaje de oro a generar (del 0 al 20%)*
76     void storeRandomDrop(std::shared_ptr<Item>& item, unsigned int goldMultiplie
77     r);
78     };
79 #endif //ARGENTUM_ITEMSFACORY_H
```


jul 21, 20 15:47	Item.h	Page 1/2
1	//	
2	// Created by agustin on 6/6/20.	
3	//	
4		
5	#ifndef ARGENTUM_ITEM_H	
6	#define ARGENTUM_ITEM_H	
7		
8	#include <string>	
9	#include <memory>	
10	#include "../libs/GameEnums.h"	
11	#include "UseReturnData.h"	
12		
13	class ItemTests;	
14	class Player;	
15		
16	//Clase interfaz de la que heredan todos los items	
17	//Los items solo tienen sentido en un inventario de un jugador	
18	class Item {	
19	private:	
20	//El id esta asociado al tipo de item que es, estos se repiten entre los di	
21	stintos	
22	//tipos de items. Ej: Se puede tener un arma y un escudo con el mismo id, pe	
23	ro se	
24	//diferencian por ser uno un arma y otro un escudo	
25	GameType::ItemType type;	
26	const std::string& name;	
27		
28	friend ItemTests;	
29	protected:	
30	int32_t id{};	
31	public:	
32	Item(GameType::ItemType _type, const std::string& name);	
33		
34	/*use debe retornar el lugar en el que debera equiparse el item una vez usad	
35	o desde	
36	un inventario, si debe ser descartado entonces se tiene que retornar INVENTO	
37	RY_PLACE_NONE	
38	Esta funcion retorna en cierta forma el tipo de item que es*/	
39	virtual GameType::EquipmentPlace use(Player& player);	
40		
41	/*Retorna el nombre del item (viene asignado desde el archivo de config)*/	
42	const std::string& getName() const ;	
43		
44	/*Esta implementada como false, el unico que la reimplementa es gold que	
45	* retorna true*/	
46	virtual bool isGold() const ;	
47		
48	/*Carga en el buffer la informacion correspondiente al item acorde	
49	* al protocolo, esta funcion se usa cuando un nuevo player se conecta	
50	* al juego*/	
51	void loadDropItemData(std::stringstream& buffer, int32_t i, int32_t j) const	
52	;	
53		
54	/*Carga en el buffer la infromacion corresopndiente al item equipado	
55	* acorde al protocolo, esta funcion se usa cuando un nuevo player	
56	* se conecta al juego*/	
57	void loadEquippedItemData(std::stringstream& buffer);	
58		
59	/*Carga en el buffer el tipo de item y su id (es decir si es un arma cual se	
60	ria,	
61	* o si es ropa cual seria. La usa el inventario para el mensaje individual	
62	* de cada player (este mensaje es para la UI personal, es decir, no para el	
63	mapa)*/	
64	void loadTypeAndId(std::stringstream& buffer);	

jul 21, 20 15:47	Item.h	Page 2/2
60		
61	virtual ~Item() = default ;	
62		
63	/*Retorna el tipo particular del item (si es una weapon retornaria que tipo	
64	* de weapon, lo mismo si fuera un chest o un shield)*/	
65	int32_t getId();	
66		
67	/*Retorna la clase del item (si es una weapon, shield, chest, etc)*/	
68	GameType::ItemType getType();	
69	};	
70		
71		
72	#endif //ARGENTUM_ITEM_H	

jul 21, 20 15:47

ItemData.h

Page 1/1

```

1  //
2  // Created by agustin on 5/7/20.
3  //
4
5  #ifndef ARGENTUM_ITEMDATA_H
6  #define ARGENTUM_ITEMDATA_H
7
8  #include <cstdint>
9  #include "../libs/GameEnums.h"
10 #include "../Map/Coordinate.h"
11
12 /*Este struct es util para encapsular los datos necesarios del item
13  * para el protocolo*/
14
15 struct ItemData {
16     GameType::ItemType type;
17     int32_t id;
18     Coordinate coordinate;
19 };
20
21 #endif //ARGENTUM_ITEMDATA_H

```

jul 21, 20 15:47

Item.cpp

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #include "Item.h"
6  #include <msgpack.hpp>
7
8  MSGPACK_ADD_ENUM(GameType::EventID)
9  MSGPACK_ADD_ENUM(GameType::ItemType)
10
11 Item::Item(GameType::ItemType _type, const std::string &_name): name(_name) {
12     type = _type;
13 }
14
15 GameType::EquipmentPlace Item::use(Player &player) {
16     return GameType::EQUIPMENT_PLACE_NONE;
17 }
18
19 const std::string &Item::getName() const {
20     return name;
21 }
22
23 bool Item::isGold() const {
24     return false;
25 }
26
27 void Item::loadDropItemData(std::stringstream &buffer, int32_t i, int32_t j) const {
28     msgpack::type::tuple<GameType::EventID> idType(GameType::CREATE_ITEM);
29     msgpack::type::tuple<GameType::ItemType, int32_t, int32_t, int32_t>
30         data(type, id, i, j);
31     msgpack::pack(buffer, idType);
32     msgpack::pack(buffer, data);
33 }
34
35 void Item::loadEquippedItemData(std::stringstream &buffer) {
36     msgpack::type::tuple<int32_t> data(id);
37     msgpack::pack(buffer, data);
38 }
39
40 void Item::loadTypeAndId(std::stringstream &buffer) {
41     msgpack::type::tuple<GameType::ItemType, int32_t> data(type, id);
42     msgpack::pack(buffer, data);
43 }
44
45 int32_t Item::getId() {
46     return id;
47 }
48
49 GameType::ItemType Item::getType() {
50     return type;
51 }

```

jul 21, 20 15:47	Inventory.h	Page 1/2
<pre> 1 // 2 // Created by agustin on 8/6/20. 3 // 4 5 #ifndef ARGENTUM_INVENTORY_H 6 #define ARGENTUM_INVENTORY_H 7 8 #include <vector> 9 #include <memory> 10 #include <unordered_map> 11 #include <list> 12 #include "Item.h" 13 #include "UseReturnData.h" 14 #include "../Server/PlayerData.hpp" 15 16 class ItemTests; 17 class EntityTests; 18 class MapTests; 19 class Game; 20 class PlayerStats; 21 class Weapon; 22 class Player; 23 class Clothing; 24 class Item; 25 class Minichat; 26 struct Coordinate; 27 28 29 /*Esta clase representa los items que almacena y tiene equipados el jugador*/ 30 class Inventory { 31 private: 32 unsigned int storedItemsAmount{}; 33 std::vector<std::shared_ptr<Item>> items; 34 std::unordered_map<GameType::EquipmentPlace, std::shared_ptr<Clothing>> clothingEquipment; 35 std::shared_ptr<Weapon> equippedWeapon; 36 37 friend ItemTests; 38 friend EntityTests; 39 friend MapTests; 40 41 private: 42 UseReturnData _manageItemPlacement(GameType::EquipmentPlace equipmentPlace, 43 unsigned int itemPosition); 44 void _dropEquippedItems(std::list<std::shared_ptr<Item>>& droppedItems); 45 static void _storeNullItemData(std::stringstream& buffer); 46 void _restoreDefaultEquipment(); 47 void _loadInitialInventory(const PlayerData& data); 48 49 public: 50 explicit Inventory(const PlayerData& data); 51 52 /*Adquiere el shared pointer recibido y lo guarda si hay espacio y retorna 53 true. Si no hay espacio o item es null no adquiere el puntero y retorna false*/ 54 bool addItem(std::shared_ptr<Item>& item); 55 56 /*Elimina el item del inventario de la posicion recibida y lo retorna, 57 dejando el lugar que ocupaba para un nuevo item que quiera ser guardado 58 Si no hay un item en la posicion retorna un shared_ptr que almacena null_ptr*/ 59 60 std::shared_ptr<Item> removeItem(unsigned int itemPosition); 61 62 /*Elimina el item con el nombre recibido del inventario y lo retorna, 63 dejando el lugar que ocupaba para un nuevo item que quiera ser guardado 64 Si no hay un item con el nombre recibido retorna un shared_ptr que almacena </pre>	<pre> 63 null_ptr*/ 64 std::shared_ptr<Item> removeItem(const std::string& itemName); 65 66 /*Usa el item en la posicion indicada, si no hay un item en la posicion no 67 hace nada*/ 68 UseReturnData useItem(Player& player, unsigned int itemPosition); 69 70 /*Retorna el danio generado por el arma dentro del rango de ella 71 * Si el target esta fuera del rango del arma retorna 0 de danio*/ 72 int getWeaponDamage(Coordinate currentPosition, Coordinate target, PlayerStats& stats) const; 73 74 /*Retorna la defensa total provista por la armadura equipada (casco, chest, 75 shield)*/ 76 unsigned int getDefense(); 77 78 /*Retorna una lista con todos los items del inventario, quitandolos del mismo*/ 79 std::list<std::shared_ptr<Item>> dropAllItems(); 80 81 /*Desequipa la ropa de la posicion seleccionada en los equipados y la almacena 82 * en el inventario. Si no tiene equipado nada (o sea un default) no hace nada*/ 83 bool unequip(GameType::EquipmentPlace clothing); 84 85 /*Desequipa el arma y la almacena en el inventario. Si no tiene equipada una 86 arma 87 * (o sea un default) no hace nada*/ 88 bool unequip(); 89 90 /*Almacena en el buffer la informacion de los items equipados del jugador 91 * acorde al protocolo*/ 92 void storeEquippedItems(std::stringstream& buffer) const; 93 94 /*Almacena toda la data relevante que tiene en el inventario/equipado el 95 * jugador acorde al protocolo. Se usa para los updates individuales 96 * que se envian periodicamente a cada player (para que sepa que tiene 97 * equipado en la UI)*/ 98 void storeAllData(std::stringstream& buffer) const; 99 100 /*Retorna el tipo de arma (Axe, Longsword, etc)*/ 101 int32_t getWeaponId(); 102 103 /*Retorna true si se tiene el item almacenado en el inventario (no equipado) 104 * false en caso contrario*/ 105 bool hasItem(const std::string& itemName); 106 107 bool isFull() const; 108 109 /*Almacena en el minichat los nombres de los items que se encuentran en el 110 * inventario*/ 111 void getInventoryNames(Minichat& chat); 112 113 /*Almacena el inventario del player en pData, se usa para el backup del archivo*/ 114 void getData(PlayerData& pData) const; 115 116 }; 117 118 #endif //ARGENTUM_INVENTORY_H </pre>	

jul 21, 20 15:47	Inventory.h	Page 2/2
	<pre> 63 null_ptr*/ 64 std::shared_ptr<Item> removeItem(const std::string& itemName); 65 66 /*Usa el item en la posicion indicada, si no hay un item en la posicion no 67 hace nada*/ 68 UseReturnData useItem(Player& player, unsigned int itemPosition); 69 70 /*Retorna el danio generado por el arma dentro del rango de ella 71 * Si el target esta fuera del rango del arma retorna 0 de danio*/ 72 int getWeaponDamage(Coordinate currentPosition, Coordinate target, PlayerStats& stats) const; 73 74 /*Retorna la defensa total provista por la armadura equipada (casco, chest, 75 shield)*/ 76 unsigned int getDefense(); 77 78 /*Retorna una lista con todos los items del inventario, quitandolos del mismo*/ 79 std::list<std::shared_ptr<Item>> dropAllItems(); 80 81 /*Desequipa la ropa de la posicion seleccionada en los equipados y la almacena 82 * en el inventario. Si no tiene equipado nada (o sea un default) no hace nada*/ 83 bool unequip(GameType::EquipmentPlace clothing); 84 85 /*Desequipa el arma y la almacena en el inventario. Si no tiene equipada una 86 arma 87 * (o sea un default) no hace nada*/ 88 bool unequip(); 89 90 /*Almacena en el buffer la informacion de los items equipados del jugador 91 * acorde al protocolo*/ 92 void storeEquippedItems(std::stringstream& buffer) const; 93 94 /*Almacena toda la data relevante que tiene en el inventario/equipado el 95 * jugador acorde al protocolo. Se usa para los updates individuales 96 * que se envian periodicamente a cada player (para que sepa que tiene 97 * equipado en la UI)*/ 98 void storeAllData(std::stringstream& buffer) const; 99 100 /*Retorna el tipo de arma (Axe, Longsword, etc)*/ 101 int32_t getWeaponId(); 102 103 /*Retorna true si se tiene el item almacenado en el inventario (no equipado) 104 * false en caso contrario*/ 105 bool hasItem(const std::string& itemName); 106 107 bool isFull() const; 108 109 /*Almacena en el minichat los nombres de los items que se encuentran en el 110 * inventario*/ 111 void getInventoryNames(Minichat& chat); 112 113 /*Almacena el inventario del player en pData, se usa para el backup del archivo*/ 114 void getData(PlayerData& pData) const; 115 116 }; 117 118 #endif //ARGENTUM_INVENTORY_H </pre>	

jul 21, 20 15:47

Shield.h

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #ifndef ARGENTUM_SHIELD_H
6  #define ARGENTUM_SHIELD_H
7
8
9  #include "Clothing.h"
10
11 class ItemTests;
12
13 /*Esta clase encapsula el comportamiento particular de los escudos*/
14
15 class Shield: public Clothing {
16 private:
17     friend ItemTests;
18
19 public:
20     explicit Shield(GameType::Clothing clothing): Clothing(clothing) {}
21
22     /*Retorna la posicion donde se equipa (como es un shield retornara la posici
on del shield)
23     * Esto es util para saber en ejecucion donde se equipa el item ya que lo gua
rdamos
24     * en el padre*/
25     GameType::EquipmentPlace use(Player& player) override;
26
27     /*Esta funcion existe para que cuando el jugador muera y tire todos sus
28     * items no tire los items default*/
29     bool isDefault() const override;
30 };
31
32
33 #endif //ARGENTUM_SHIELD_H

```

jul 21, 20 15:47

Shield.cpp

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #include "Shield.h"
6
7  GameType::EquipmentPlace Shield::use(Player &player) {
8      return GameType::EQUIPMENT_PLACE_SHIELD;
9  }
10
11 bool Shield::isDefault() const {
12     return (static_cast<GameType::Clothing>(id) == GameType::Clothing::NO_SHIELD)
13     ;
14 }

```

jul 21, 20 15:47

Head.h

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #ifndef ARGENTUM_HEAD_H
6  #define ARGENTUM_HEAD_H
7
8
9  #include "Clothing.h"
10
11 class ItemTests;
12
13 /*Esta clase encapsula el comportamiento particular de la ropa que se
14  * equipa en la cabeza (cascos, sombreros)*/
15
16 class Head: public Clothing {
17 private:
18     friend ItemTests;
19
20 public:
21     explicit Head(GameType::Clothing clothing) : Clothing(clothing) {}
22
23     /*Retorna la posicion donde se equipa (como es un head retornara la posicion
24     del head)
25     * Esto es util para saber en ejecucion donde se equipa el item ya que lo gua
26     rdamos
27     * en el padre*/
28     GameType::EquipmentPlace use(Player& player) override;
29
30     /*Esta funcion existe para que cuando el jugador muera y tire todos sus
31     * items no tire los items default*/
32     bool isDefault() const override;
33 };
34 #endif //ARGENTUM_HEAD_H

```

jul 21, 20 15:47

Head.cpp

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #include "Head.h"
6
7  GameType::EquipmentPlace Head::use(Player &player) {
8      return GameType::EQUIPMENT_PLACE_HEAD;
9  }
10
11 bool Head::isDefault() const {
12     return (static_cast<GameType::Clothing>(id) == GameType::Clothing::NO_HELMET)
13 ;
14 }

```

jul 21, 20 15:47

Clothing.h

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #ifndef ARGENTUM_CLOTHING_H
6  #define ARGENTUM_CLOTHING_H
7
8  #include "../Item.h"
9  #include "../Config/Configuration.h"
10
11 class ItemTests;
12
13 /*Esta clase encapsula el comportamiento general de la ropa*/
14
15 class Clothing : public Item {
16 private:
17     unsigned int minDefense;
18     unsigned int maxDefense;
19
20     friend ItemTests;
21
22 public:
23     explicit Clothing(GameType::Clothing clothing);
24
25     /*Retorna un random entre la minima y maxima defensa del item*/
26     unsigned int getDefense() const;
27
28     /*Esta funcion existe para que cuando el player muera y tire todos sus items
29     sepa reconocer los default y no los tire*/
30     virtual bool isDefault() const = 0;
31
32     virtual ~Clothing() = default;
33 };
34
35
36 #endif //ARGENTUM_CLOTHING_H

```

jul 21, 20 15:47

Clothing.cpp

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #include "Clothing.h"
6  #include "../Config/Calculator.h"
7
8  Clothing::Clothing(GameType::Clothing clothing): Item(GameType::ITEM_TYPE_CLOTHI
NG,
9              Configuration::getInstance().configClothingData(clothing).name)
10 {
11     Config::ClothingData stats = Configuration::getInstance().configClothingData
(clothing);
12     id = static_cast<unsigned int>(clothing);
13     minDefense = stats.minDefense;
14     maxDefense = stats.maxDefense;
15 }
16
17 unsigned int Clothing::getDefense() const {
18     return Calculator::getRandomInt(static_cast<int>(minDefense),
19                                     static_cast<int>(maxDefense));
20 }
21

```

jul 21, 20 15:47

Chest.h

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #ifndef ARGENTUM_CHEST_H
6  #define ARGENTUM_CHEST_H
7
8  #include "Clothing.h"
9
10 class ItemTests;
11
12 /*Esta clase encapsula el comportamiento particular de la ropa equipable
13  * en el chest*/
14
15 class Chest: public Clothing {
16 private:
17     friend ItemTests;
18
19 public:
20     explicit Chest(GameType::Clothing clothing) : Clothing(clothing) {}
21
22     /*Retorna la posicion donde se equipa (como es un chest retornara la posicio
23     n del chest)
24     * Esto es util para saber en ejecucion donde se equipa el item ya que lo gu
25     ardamos
26     * en el padre*/
27     GameType::EquipmentPlace use(Player& player) override;
28
29     /*Esta funcion existe para que cuando el jugador muera y tire todos sus
30     items no tire los items default*/
31     bool isDefault() const override;
32 };
33 #endif //ARGENTUM_CHEST_H

```

jul 21, 20 15:47

Chest.cpp

Page 1/1

```

1  //
2  // Created by agustin on 9/6/20.
3  //
4
5  #include "Chest.h"
6
7  GameType::EquipmentPlace Chest::use(Player &player) {
8      return GameType::EQUIPMENT_PLACE_CHEST;
9  }
10
11 bool Chest::isDefault() const {
12     return (id == GameType::COMMON_CLOTHING);
13 }

```

jul 21, 20 15:47	Weapon.h	Page 1/1
<pre> 1 // 2 // Created by agustin on 6/6/20. 3 // 4 5 #ifndef ARGENTUM_WEAPON_H 6 #define ARGENTUM_WEAPON_H 7 8 #include "../Item.h" 9 #include "../libs/GameEnums.h" 10 11 class ItemTests; 12 class PlayerStats; 13 struct Coordinate; 14 15 /*Esta clase encapsula el comportamiento de las armas*/ 16 17 class Weapon : public Item { 18 private: 19 int minDamage{}; 20 int maxDamage{}; 21 unsigned int attackRange{}; 22 int manaConsumption{}; 23 24 friend ItemTests; 25 26 private: 27 bool _isTargetReachable(Coordinate attackPosition, Coordinate attackedPositi on) const; 28 void _initializeData(int minDamage, int maxDamage, 29 unsigned int _manaConsumption, unsigned int _range); 30 31 public: 32 explicit Weapon(GameType::Weapon weapon); 33 34 /*Devuelve el danio que haria el arma por sã sola, es un numero aleatorio 35 entre el danio minimo y el maximo 36 Recibe las posiciones para determinar si el ataque es realizado o no, si 37 no lo realiza retorna 0 38 Podria recibir la distancia, pero esto permite encapsular la forma de 39 calcular la distancia en la Weapon*/ 40 int getDamage(Coordinate attackPosition, Coordinate attackedPosition, Player Stats& stats) const; 41 42 /*Retorna la posicion donde se equipa (como esta es un arma sera en el lugar 43 del arma) 44 * Esto es util para saber en ejecucion donde se equipa el item ya que lo gu ardamos 45 * en el padre*/ 46 GameType::EquipmentPlace use(Player& player) override; 47 48 /*Esta funcion existe para que cuando el jugador muera y tire todos sus 49 items no tire los items default*/ 50 bool isDefault(); 51 52 ~Weapon() override; 53 }; 54 55 #endif //ARGENTUM_WEAPON_H </pre>		

jul 21, 20 15:47	Weapon.cpp	Page 1/1
<pre> 1 // 2 // Created by agustin on 6/6/20. 3 // 4 5 #include "Weapon.h" 6 #include "../Config/Calculator.h" 7 #include "../Map/Coordinate.h" 8 #include "../Config/Configuration.h" 9 #include "../Entities/PlayerStats.h" 10 #include <ctime> 11 #include <cstdlib> 12 13 //////////////////////////////////////PRIVATE//////////////////////////////////// 14 15 bool Weapon::_isTargetReachable(Coordinate attackPosition, 16 Coordinate attackedPosition) const { 17 unsigned int distance = std::abs(attackPosition.iPosition - attackedPosition .iPosition) + 18 std::abs(attackPosition.jPosition - attackedPosition .jPosition); 19 return (distance != 0) ^ (distance <= attackRange); 20 } 21 22 23 void Weapon::_initializeData(int _minDamage, int _maxDamage, unsigned int _manaC onsumption, 24 unsigned int _range) { 25 minDamage = _minDamage; 26 maxDamage = _maxDamage; 27 manaConsumption = static_cast<int>(_manaConsumption); 28 attackRange = _range; 29 } 30 31 32 //////////////////////////////////////PUBLIC//////////////////////////////////// 33 Weapon::Weapon(GameType::Weapon weapon): Item(GameType::ITEM_TYPE_WEAPON, 34 Configuration::getInstance().confi gWeaponData(weapon).name) { 35 Config::WeaponData stats = Configuration::getInstance().configWeaponData(wea pon); 36 id = static_cast<unsigned int>(weapon); 37 _initializeData(stats.minDmg, stats.maxDmg, stats.manaConsumption, stats.ran ge); 38 } 39 40 int Weapon::getDamage(Coordinate attackPosition, Coordinate attackedPosition, Pl ayerStats& stats) const { 41 if (!_isTargetReachable(attackPosition, attackedPosition) & ~stats.consumeM ana(manaConsumption)) { 42 return 0; 43 } 44 return Calculator::getRandomInt(minDamage, maxDamage); 45 } 46 47 GameType::EquipmentPlace Weapon::use(Player &player) { 48 return GameType::EQUIPMENT_PLACE_WEAPON; 49 } 50 51 Weapon::~Weapon() = default; 52 53 bool Weapon::isDefault() { 54 return (static_cast<GameType::Weapon>(id) == GameType::Weapon::FIST); 55 } 56 </pre>		

jul 21, 20 15:47

ShouldPlayerBeRevived.h

Page 1/1

```

1 //
2 // Created by agustin on 8/7/20.
3 //
4
5 #ifndef ARGENTUM_SHOULDPLAYERBEREVIVED_H
6 #define ARGENTUM_SHOULDPLAYERBEREVIVED_H
7
8 #include <sstream>
9 #include "../Map/Coordinate.h"
10
11 struct ResurrectData;
12 class Map;
13
14 //Functor que indica cuando un jugador debe ser eliminado de la lista de jugadores
15 //a revivir, se debe pasar a remove_if para la lista de monsters
16 //Almacena en la lista recibida las coordenadas de los monsters que se eliminaron para
17 //sacarlos despues del mapa
18 class ShouldPlayerBeRevived {
19 private:
20     Map& map;
21     std::stringstream& data;
22     double timeStep;
23
24 private:
25     void _storeResurrectMessage(const ResurrectData& resurrectData);
26     void _storeTeleportMessage(const ResurrectData& resurrectData, Coordinate newPosition);
27
28 public:
29     explicit ShouldPlayerBeRevived(Map& map, std::stringstream& data, double timeStep);
30
31     bool operator()(ResurrectData& resurrectData);
32 };
33
34
35 #endif //ARGENTUM_SHOULDPLAYERBEREVIVED_H

```

jul 21, 20 15:47

ShouldPlayerBeRevived.cpp

Page 1/1

```

1 //
2 // Created by agustin on 8/7/20.
3 //
4
5 #include "ShouldPlayerBeRevived.h"
6
7 #include "ResurrectData.h"
8 #include "../libs/GameEnums.h"
9 #include "../Entities/Player.h"
10 #include "../Map/Map.h"
11 #include <msgpack.hpp>
12
13 MSGPACK_ADD_ENUM(GameType::EventID)
14
15 ShouldPlayerBeRevived::ShouldPlayerBeRevived(Map& map, std::stringstream& data, double _timeStep)
16     : map(map), data(data) {
17     timeStep = _timeStep;
18 }
19
20 bool ShouldPlayerBeRevived::operator()(ResurrectData& resurrectData) {
21     if (!resurrectData.playerToResurrect->isDead()) {
22         return true;
23     }
24     resurrectData.timeWaited += timeStep;
25     Coordinate noFreePositionReturn = {-1, -1};
26     Coordinate positionToTeleport{};
27     if (resurrectData.timeWaited ≥ resurrectData.timeToWait) {
28         positionToTeleport = map.getSpawnCoordinateArroundPosition(resurrectData.resurrectingPriest);
29         if (positionToTeleport ≠ noFreePositionReturn) {
30             map.moveEntity(resurrectData.playerToResurrect->getPosition(), positionToTeleport);
31             resurrectData.playerToResurrect->resetMovement();
32             resurrectData.playerToResurrect->restoreStats(true);
33             _storeResurrectMessage(resurrectData);
34             _storeTeleportMessage(resurrectData, positionToTeleport);
35             return true;
36         }
37     }
38     return false;
39 }
40
41 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
42
43 void ShouldPlayerBeRevived::_storeResurrectMessage(const ResurrectData& resurrectData) {
44     msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::RESURRECTED);
45     msgpack::pack(data, messageTypeData);
46     msgpack::type::tuple<std::string> resurrectDataTuple(resurrectData.playerToResurrect->getNickname());
47     msgpack::pack(data, resurrectDataTuple);
48 }
49
50 void ShouldPlayerBeRevived::_storeTeleportMessage(const ResurrectData& resurrectData, Coordinate newPosition) {
51     msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::TELEPORTED);
52     msgpack::pack(data, messageTypeData);
53     msgpack::type::tuple<std::string, int32_t, int32_t> resurrectDataTuple(resurrectData.playerToResurrect->getNickname(), newPosition.iPosition, newPosition.jPosition);
54     msgpack::pack(data, resurrectDataTuple);
55 }
56

```

jul 21, 20 15:47

ShouldMonsterBeRemoved.h

Page 1/1

```

1  //
2  // Created by agustin on 21/6/20.
3  //
4
5  #ifndef ARGENTUM_SHOULDMONSTERBEREMOVED_H
6  #define ARGENTUM_SHOULDMONSTERBEREMOVED_H
7
8  #include <list>
9  #include <memory>
10 #include "../Map/Coordinate.h"
11
12 class Monster;
13
14 //Functor que indica cuando un monstruo debe ser eliminado de la lista, se debe
  pasar
15 //a remove_if para la lista de monsters
16 //Almacena en la lista recibida las coordenadas de los monsters que se eliminara
  n para
17 //sacarlos despues del mapa
18 class ShouldMonsterBeRemoved {
19 private:
20     std::list<std::pair<Coordinate, const std::string*>>& monstersToRemove;
21 public:
22     explicit ShouldMonsterBeRemoved(std::list<std::pair<Coordinate, const std::s
  tring*>>& monstersToRemove);
23
24     //Retorna true si el monstruo esta muerto y guarda su coordenada en la lista
25     //almacenada
26     bool operator()(const Monster* monster);
27 };
28
29 #endif //ARGENTUM_SHOULDMONSTERBEREMOVED_H

```

jul 21, 20 15:47

ShouldMonsterBeRemoved.cpp

Page 1/1

```

1  //
2  // Created by agustin on 21/6/20.
3  //
4
5  #include "ShouldMonsterBeRemoved.h"
6
7  #include "../Entities/Monster.h"
8
9  ShouldMonsterBeRemoved::ShouldMonsterBeRemoved(std::list<std::pair<Coordinate, c
  onst std::string*>> &monstersToRemove):
10      monstersToRemove(monstersToRemove)
11  {
12  }
13
14  bool ShouldMonsterBeRemoved::operator()(const Monster* monster) {
15      if (monster->isDead()) {
16          std::pair<Coordinate, const std::string*> aux(monster->getPosition(), &(
  monster->getNickname()));
17          monstersToRemove.push_back(std::move(aux));
18          return true;
19      }
20      return false;
21  }

```

jul 21, 20 15:47

ResurrectData.h

Page 1/1

```

1  //
2  // Created by agustin on 8/7/20.
3  //
4
5  #ifndef ARGENTUM_RESURRECTDATA_H
6  #define ARGENTUM_RESURRECTDATA_H
7
8  #include "../Map/Coordinate.h"
9
10 class Player;
11
12 //struct auxiliar para guardar la informacion de un player que sera resucitado
13 struct ResurrectData {
14     double timeToWait;
15     double timeWaited;
16     Coordinate resurrectingPriest;
17     Player* playerToResurrect;
18 };
19
20 #endif //ARGENTUM_RESURRECTDATA_H

```

jul 21, 20 15:47

MonstersFactory.h

Page 1/1

```

1  //
2  // Created by agustin on 20/6/20.
3  //
4
5  #ifndef ARGENTUM_MONSTERSFACTORY_H
6  #define ARGENTUM_MONSTERSFACTORY_H
7
8  #include <unordered_map>
9  #include <vector>
10 #include <memory>
11 #include "../libs/GameEnums.h"
12 #include "../Map/Coordinate.h"
13
14 class GameTests;
15 class Monster;
16 class Game;
17 class Map;
18
19 typedef void (*monsterCreator)(Game& game, Coordinate initialPosition,
20                                std::shared_ptr<Monster>& monster);
21
22 //Esta clase se encarga de crear un monstruo aleatorio, se utiliza para realizar
23 //los spawns aleatorios de monsters
24 class MonstersFactory {
25 private:
26     std::unordered_map<GameType::Entity, monsterCreator> monsterCreators;
27     std::vector<GameType::Entity> existingMonsters;
28
29     friend GameTests;
30
31 private:
32     static void _storeSpider(Game& game, Coordinate initialPosition,
33                              std::shared_ptr<Monster>& monster);
34     static void _storeSkeleton(Game& game, Coordinate initialPosition,
35                                std::shared_ptr<Monster>& monster);
36     static void _storeZombie(Game& game, Coordinate initialPosition,
37                               std::shared_ptr<Monster>& monster);
38     static void _storeGoblin(Game& game, Coordinate initialPosition,
39                              std::shared_ptr<Monster>& monster);
40
41 public:
42     MonstersFactory();
43
44     //Guarda en monster un monstruo aleatorio, la coordenada inicial es el {0, 0
45     void storeRandomMonster(Game& game, std::shared_ptr<Monster>& monster);
46 };
47
48 #endif //ARGENTUM_MONSTERSFACTORY_H

```

jul 21, 20 15:47

MonstersFactory.cpp

Page 1/1

```

1  //
2  // Created by agustin on 20/6/20.
3  //
4
5
6
7  #include "MonstersFactory.h"
8
9  #include "../Entities/Monster.h"
10 #include "../Config/Calculator.h"
11
12
13 //////////////////////////////////PRIVATE////////////////////////////////////
14
15 void MonstersFactory::_storeSpider(Game& game, Coordinate initialPosition, std::
shared_ptr<Monster>& monster) {
16     monster = std::make_shared<Monster>(game, initialPosition, GameType::SPIDER,
GameType::SPIDER_ATTACK);
17 }
18
19 void MonstersFactory::_storeSkeleton(Game& game, Coordinate initialPosition, std
::shared_ptr<Monster>& monster) {
20     monster = std::make_shared<Monster>(game, initialPosition, GameType::SKELETO
N, GameType::SKELETON_ATTACK);
21 }
22
23 void MonstersFactory::_storeZombie(Game& game, Coordinate initialPosition, std::s
hared_ptr<Monster>& monster) {
24     monster = std::make_shared<Monster>(game, initialPosition, GameType::ZOMB
IE, GameType::ZOMBIE_ATTACK);
25 }
26
27 void MonstersFactory::_storeGoblin(Game& game, Coordinate initialPosition, std::
shared_ptr<Monster>& monster) {
28     monster = std::make_shared<Monster>(game, initialPosition, GameType::GOBLIN,
GameType::GOBLIN_ATTACK);
29 }
30
31 //////////////////////////////////PUBLIC////////////////////////////////////
32
33 MonstersFactory::MonstersFactory() {
34     monsterCreators[GameType::SPIDER] = _storeSpider;
35     monsterCreators[GameType::SKELETON] = _storeSkeleton;
36     monsterCreators[GameType::ZOMBIE] = _storeZombie;
37     monsterCreators[GameType::GOBLIN] = _storeGoblin;
38
39     for (const auto & creator: monsterCreators) {
40         existingMonsters.push_back(creator.first);
41     }
42 }
43
44 void MonstersFactory::storeRandomMonster(Game& game, std::shared_ptr<Monster> &m
onster) {
45     //ejecuta la funcion que se almacena en el unordered_map con la key que corr
esponde a la
46     //posicion aleatoria del vector de keys
47     monsterCreators[existingMonsters[Calculator::getRandomInt(0, static_cast<int
>(existingMonsters.size()) - 1)]]
48         (game, {0, 0}, monster);
49 }
50
51

```

jul 21, 20 15:47

Game.h

Page 1/3

```

1  //
2  // Created by agustin on 7/6/20.
3  //
4
5  #ifndef ARGENTUM_GAME_H
6  #define ARGENTUM_GAME_H
7
8  #include <memory>
9  #include <queue>
10 #include "../Map/Map.h"
11 #include "MonstersFactory.h"
12 #include "Events/Event.h"
13 #include "../Items/ItemData.h"
14 #include "ResurrectData.h"
15 #include "../libs/Timer.h"
16
17 class EntityTests;
18 struct PlayerData;
19
20 class PlayerShouldBeRemoved {
21 private:
22     Player* playerToRemove;
23
24 public:
25     explicit PlayerShouldBeRemoved(Player* player) : playerToRemove(player) {}
26     bool operator()(const Player* player);
27 };
28
29 struct MoveCommand {
30     Coordinate initialPosition;
31     Coordinate finalPosition;
32     bool isTeleporting;
33 };
34
35 //Esta clase se encarga de manejar en forma general las acciones que quiere real
izar
36 //cada identidad
37 class Game {
38 private:
39     std::list<Coordinate> priests;
40     Map map;
41     std::queue<std::unique_ptr<Event>> eventQueue;
42
43     unsigned int monsterCreationRate;
44     unsigned int maxNumberOfMonsters;
45     unsigned int spawnInterval;
46     Timer monsterSpawnTimer;
47     MonstersFactory monstersFactory;
48
49     std::list<Monster*> monsters;
50     std::unordered_map<std::string, Player*> players;
51     std::unordered_map<Coordinate, const Item*> mapItems;
52
53     std::list<ResurrectData> playersToResurrect;
54
55     friend GameTests;
56     friend EntityTests;
57
58 private:
59
60     void _removeMonsters(ServerProtocol& protocol);
61     void _updateMonsters(double timeStep);
62     void _updatePlayers(double timeStep);
63     void _executeQueueOperations(ServerProtocol& protocol);
64     void _repopulateMap(ServerProtocol& protocol);
65

```

jul 21, 20 15:47	Game.h	Page 2/3
66	void _updateDeadPlayersTimer(ServerProtocol& protocol, double timestep);	
67		
68	public:	
69		
70	//Este constructor debe ser utilizado unicamente para las pruebas	
71	//explicit Game(ClientsMonitor&& clientAux = ClientsMonitor()){};	
72		
73	explicit Game(MapFileReader^ mapFile);	
74		
75	//Delega a map el ataque a la coordenada recibida, retorna una instancia de	
76	AttackResult junto	
77	//con un bool que esta en true si se realizo un ataque, sino retorna false	
78	std::pair<AttackResult, bool> attackPosition(int damage, unsigned int level,	
79	bool isAPlayer,	
80	Coordinate coordinate);	
81		
82	//Llama a Map para que guarde los items recibidos en el tile que corresponde	
83	//a la coordenada recibida	
84	void dropItems(std::list<std::shared_ptr<Item>>^ items, Coordinate position	
85);	
86		
87	//Llama a Map para que guarde el item recibido en el tile que corresponde	
88	//a la coordenada recibida	
89	void dropItems(std::shared_ptr<Item>^ item, Coordinate position);	
90		
91	//Retorna una referencia constante del mapa, util para los monstruos	
92	const Map& getMap() const;	
93		
94	//Ejecuta un update del juego, realiza todas las acciones que debe realizar	
95	el juego	
96	//en este tick	
97	void update(double timeStep, ServerProtocol& protocol);	
98		
99	//Delega el comportamiento a la entity que guarda, si es que guarda una	
100	//unsigned int list(Player& player, std::list<ProductData>& products, Coordi	
101	nate coordinate);	
102	void list(Player& player, Coordinate coordinate);	
103		
104	//Delega el comportamiento a la entity que guarda, si es que guarda una	
105	void withdraw(Player& player, const std::string& itemName, Coordinate coordi	
106	nate);	
107		
108	//Delega el comportamiento a la entity que guarda, si es que guarda una	
109	void deposit(Player& player, const std::string& itemName, Coordinate coordin	
110	ate);	
111		
112	//Delega el comportamiento a la entity que guarda, si es que guarda una	
113	void buy(Player& player, const std::string& itemName, Coordinate coordinate)	
114	;	
115		
116	//Delega el comportamiento a la entity que guarda, si es que guarda una	
117	void sell(Player& player, const std::string& itemName, Coordinate coordinate	
118);	
119		
120	//Delega a map el movimiento de la entidad que se encuentra en initialPositi	
121	on	
122	//y debe terminar en finalPosition	
123	void moveEntity(Coordinate initialPosition, Coordinate finalPosition);	
124		
125	//Game se apropia del puntero al evento, agregandolo a la cola de enentos qu	
126	e	
127	//despues sera vaciada para ejecutar las acciones del update. Este metodo es	
128	//virtual para que Fakeit pueda redefinirlo a que no haga nada porque sino l	
129	a	
130	//prueba segmentea (el stub no reserva memoria para la cola de eventos creem	
131	os)	

jul 21, 20 15:47	Game.h	Page 3/3
119	virtual void pushEvent(std::unique_ptr<Event>^ event);	
120		
121	//Crea el player en base al nickname, raza y clase que recibe	
122	Player& createPlayer(PlayerData& playerData, ServerProtocol& protocol);	
123		
124	//Carga en el protocolo el estado actual del juego para mandar a un jugador	
125	//que se conecta y poder mandarle luego solo los cambios en cada update	
126	const std::vector<char>& getCurrentState(ServerProtocol& protocol);	
127		
128	//Elimina al jugador del juego, eliminandolo de todos los lugares en los que	
129	//esta guardado, guarda en el protocolo el mensaje que comunica al resto de	
130	los	
131	//clientes que desaparecio un jugador	
132	void removePlayer(const std::string& playerNickname, ServerProtocol& protoco	
133	l);	
134		
135	//Intenta guardar el item en el inventario del player, retorna el puntero al	
136	//item que se encuentra al final de la lista de items guardada en el tile de	
137	l cual	
138	//se agarro el item, si no hay mas items despues de agarrar uno entonces ret	
139	orna nullptr	
140	const Item* storeItemFromTileInPlayer(Player& player);	
141		
142	//Resuscita el player instantaneamente si la coordenada que selecciono contie	
143	ne un priest, sino	
144	//guarda su informacion en la lista de players a resucitar para resucitarlo	
145	cuando pase la cantidad	
146	//de tiempo necesaria	
147	//Si lo resucita instantaneamente retorna true, sino retorna false	
148	bool requestResurrect(Player& player, Coordinate selectedPosition);	
149		
150	//Envia un mensaje a otro jugador	
151	void messagePlayer(Player& playerWhoMessaged, const std::string& playerToMes	
152	sage, const std::string& message);	
153		
154		
155	//Retorna true si el player se encuentra en el game, false en caso contrario	
156	bool playerExists(const std::string& nickname) const;	
157		
158	//Delega a map el pedido del player al entity de la coordenada target para	
159	//que se restore su mana y vida	
160	void requestRestore(Player& player, Coordinate target);	
161		
162		
163	//Guarda en playerData los items que tiene guardados en el banker el player	
164	//que tiene el nickname guardado en playerData	
165	static void getPlayerBank(PlayerData& playerData) ;	
166		
167	};	
168		
169		
170		
171	#endif //ARGENTUM_GAME_H	

jul 21, 20 15:47

Game.cpp

Page 1/5

```

1  //
2  // Created by agustin on 7/6/20.
3  //
4
5  #include <algorithm>
6  #include "Game.h"
7  #include "../Entities/AttackResult.h"
8  #include "../Entities/Monster.h"
9  #include "ShouldMonsterBeRemoved.h"
10 #include "Events/Event.h"
11 #include "../Server/ServerProtocol.h"
12 #include "../Entities/Player.h"
13 #include "ShouldPlayerBeRevived.h"
14 #include <iostream>
15 #include "../Server/PlayerData.hpp"
16 #include "../Entities/Citizens/Banker.h"
17 #include "../Config/Configuration.h"
18
19 MSGPACK_ADD_ENUM(GameType::EventID)
20
21 #define WAITING_TIME_MESSAGE "The estimated waiting time to resurrect is "
22 #define NO_PLAYER_MESSAGE "That player does not exist or is not connected\n"
23 #define WELCOME_MESSAGE "Welcome to Argentum traveller!\nWe hope you enjoy our work\nSincerely, AIM Team\n"
24 const Coordinate defaultSpawnPoint = {88,83};
25
26
27 //////////////////////////////////PRIVATE////////////////////////////////////
28
29 //Carga hasta monsterCreationRate monstruos nuevos cada cierto intervalo de tiempo
30 //Si la cantidad que se desea crear sobrepasa la cantidad maxima, entonces crea hasta
31 //conseguir la cantidad maxima
32 void Game::_repopulateMap(ServerProtocol& protocol) {
33     Coordinate aux{};
34     std::stringstream data;
35     double timePassed = monsterSpawnTimer.getTime();
36     if (timePassed > spawnInterval) {
37         unsigned int monstersToCreate = monsterCreationRate;
38         std::shared_ptr<Monster> monster;
39         monsterSpawnTimer.start();
40         if ((monstersToCreate + monsters.size()) > maxNumberOfMonsters) {
41             monstersToCreate = maxNumberOfMonsters - monsters.size();
42         }
43         for (unsigned int i = 0; i < monstersToCreate; ++i) {
44             monstersFactory.storeRandomMonster(*this, monster);
45             aux = map.getMonsterCoordinate();
46             monster->setPosition(aux);
47             (*monster) >> data;
48             monsters.push_back(monster.get());
49             map.addEntity(aux, std::static_pointer_cast<Entity>(monster));
50         }
51         protocol.addToGeneralData(data);
52     }
53 }
54
55 //Vacía la cola de operaciones a realizar, ejecutando cada operacion que es
56 //desencolada
57 void Game::_executeQueueOperations(ServerProtocol& protocol) {
58     while (!eventQueue.empty()) {
59         (*eventQueue.front())(protocol);
60         eventQueue.pop();
61     }
62 }
63

```

jul 21, 20 15:47

Game.cpp

Page 2/5

```

64
65 void Game::moveEntity(Coordinate initialPosition, Coordinate finalPosition) {
66     map.moveEntity(initialPosition, finalPosition);
67 }
68
69 //Llama a update de todos los monstruos que se encuentran en el mapa, haciendo
70 //que tomen una decision
71 void Game::_updateMonsters(double timeStep) {
72     for (const auto & monster: monsters) {
73         monster->update(timeStep);
74     }
75 }
76
77 //Hace un update de los players conectados, actualizando su vida, mana y distancia
78 //recorrida
79 void Game::_updatePlayers(double timeStep) {
80     for (const auto & player: players) {
81         player.second->update(timeStep);
82     }
83 }
84
85 //Elimina de las listas almacenadas y del mapa los monsters que deban ser eliminados
86 //y guarda en el protocolo el mensaje de que estos deben desaparecer para mandar al cliente
87 void Game::_removeMonsters(ServerProtocol& protocol) {
88     std::stringstream data;
89     std::list<std::pair<Coordinate, const std::string*>> monstersToRemove;
90     ShouldMonsterBeRemoved shouldBeRemoved(monstersToRemove);
91     monsters.erase(std::remove_if(monsters.begin(), monsters.end(), shouldBeRemoved), monsters.end());
92     for (const auto & monster: monstersToRemove) {
93         msgpack::type::tuple<GameType::EventID> eventIdData(GameType::EventID::REMOVE_ENTITY);
94         msgpack::pack(data, eventIdData);
95         msgpack::type::tuple<std::string> removedMonsterNickname(*monster.second);
96         msgpack::pack(data, removedMonsterNickname);
97         protocol.addToGeneralData(data);
98         map.removeEntity(monster.first);
99     }
100 }
101
102 //Itera la lista de players muertos que estan esperando resucitar, aplicando la funcion
103 //shouldBeRevived y eliminando asi los players que son revividos
104 void Game::_updateDeadPlayersTimer(ServerProtocol& protocol, double timestep) {
105     std::stringstream data;
106     ShouldPlayerBeRevived shouldBeRevived(map, data, timestep);
107     playersToResurrect.erase(std::remove_if(playersToResurrect.begin(), playersToResurrect.end(), shouldBeRevived), playersToResurrect.end());
108     protocol.addToGeneralData(data);
109 }
110
111
112
113
114 //////////////////////////////////PUBLIC////////////////////////////////////
115
116 std::pair<AttackResult, bool> Game::attackPosition(int damage, unsigned int level, bool isAPlayer,
117                                                     Coordinate coordinate) {
118     return map.attackTile(damage, level, isAPlayer, coordinate);
119 }
120
121 void Game::dropItems(std::list<std::shared_ptr<Item>> items, Coordinate position) {

```

jul 21, 20 15:47

Game.cpp

Page 3/5

```

on) {
122     if (items.empty()) {
123         throw std::invalid_argument("Received empty list in Game::dropItems");
124     }
125     mapItems[position] = items.back().get();
126     map.addToTile(std::move(items), position);
127 }
128
129 void Game::dropItems(std::shared_ptr<Item> ^item, Coordinate position) {
130     if (!item) {
131         throw std::invalid_argument("Received null item in Game::dropItems");
132     }
133     mapItems[position] = item.get();
134     map.addToTile(std::move(item), position);
135 }
136
137 void Game::update(double timeStep, ServerProtocol& protocol) {
138     _repopulateMap(protocol);
139     _updateMonsters(timeStep);
140     _updatePlayers(timeStep);
141     _executeQueueOperations(protocol);
142     _removeMonsters(protocol);
143     _updateDeadPlayersTimer(protocol, timeStep);
144 }
145
146 Game::Game(MapFileReader^ mapFile): priests(), map(mapFile, priests) {
147     Configuration& config = Configuration::getInstance();
148     monsterCreationRate = config.configMonsterSpawnAmount();
149     maxNumberOfMonsters = config.configMaxMonsterAmount();
150     spawnInterval = config.configTimeBetweenMonsterSpawns();
151     monsterSpawnTimer.start();
152 }
153
154 const Map& Game::getMap() const {
155     return map;
156 }
157
158 void Game::list(Player &player, Coordinate coordinate) {
159     map.list(player, coordinate);
160 }
161
162 void Game::withdraw(Player &player, const std::string &itemName, Coordinate coordinate) {
163     map.withdraw(player, itemName, coordinate);
164 }
165
166 void Game::deposit(Player &player, const std::string &itemName, Coordinate coordinate) {
167     map.deposit(player, itemName, coordinate);
168 }
169
170 void Game::buy(Player &player, const std::string &itemName, Coordinate coordinate) {
171     map.buy(player, itemName, coordinate);
172 }
173
174 void Game::sell(Player &player, const std::string &itemName, Coordinate coordinate) {
175     map.sell(player, itemName, coordinate);
176 }
177
178 void Game::pushEvent(std::unique_ptr<Event>^ event) {
179     eventQueue.push(std::move(event));
180 }
181
182 Player& Game::createPlayer(PlayerData& playerData, ServerProtocol& protocol) {

```

jul 21, 20 15:47

Game.cpp

Page 4/5

```

183     Coordinate spawnPosition{};
184     if (!priests.empty()) {
185         spawnPosition = map.getSpawnCoordinateArroundPosition(priests.front());
186     } else {
187         spawnPosition = map.getSpawnCoordinateArroundPosition(defaultSpawnPoint);
188     }
189     Banker::addPlayerItems(playerData);
190     auto player = std::make_shared<Player>(*this, spawnPosition, playerData);
191     Player* playerAux = player.get();
192     players.emplace(playerAux->getNickname(), playerAux);
193     player->addMessage(WELCOME_MESSAGE);
194     map.addEntity(spawnPosition, std::move(player));
195     std::stringstream data;
196     (*playerAux) >> data;
197     protocol.addToGeneralData(data);
198     return *playerAux;
199 }
200
201 const std::vector<char>& Game::getCurrentState(ServerProtocol& protocol) {
202     return protocol.buildCurrentState(players, monsters, mapItems);
203 }
204
205 void Game::removePlayer(const std::string& playerNickname, ServerProtocol& protocol) {
206     std::stringstream data;
207     msgpack::type::tuple<GameType::EventID> eventIdData(GameType::EventID::REMOVE_ENTITY);
208     msgpack::pack(data, eventIdData);
209     msgpack::type::tuple<std::string> removedPlayerNickname(playerNickname);
210     msgpack::pack(data, removedPlayerNickname);
211     protocol.addToGeneralData(data);
212     Coordinate playerPosition = players.at(playerNickname)->getPosition();
213     players.erase(playerNickname);
214     Banker::erasePlayerItems(playerNickname);
215     map.removeEntity(playerPosition);
216 }
217
218 const Item* Game::storeItemFromTileInPlayer(Player& player) {
219     Coordinate playerPosition = player.getPosition();
220     std::shared_ptr<Item> retrievedItem = map.removeItem(playerPosition);
221     const Item* returnData = nullptr;
222     if (retrievedItem) {
223         if (!player.storeItem(retrievedItem)) {
224             returnData = retrievedItem.get();
225             map.addToTile(std::move(retrievedItem), playerPosition);
226         } else {
227             returnData = map.peekShowedItemData(playerPosition);
228             if (returnData) {
229                 mapItems[{playerPosition.iPosition, playerPosition.jPosition}] =
230                     returnData;
231             } else {
232                 mapItems.erase({playerPosition.iPosition, playerPosition.jPosition});
233             }
234         }
235     }
236     return returnData;
237 }
238
239 bool Game::requestResurrect(Player &player, Coordinate selectedPosition) {
240     if (priests.empty() || !player.isDead()) {
241         return false;
242     }
243     Coordinate playerPosition = player.getPosition();

```

jul 21, 20 15:47

Game.cpp

Page 5/5

```

244     Coordinate nearestPriest = priests.front();
245     for (const auto & priestPosition: priests) {
246         if (selectedPosition == priestPosition) {
247             player.restoreStats(true);
248             return true;
249         }
250         if (playerPosition.calculateDistance(priestPosition) <
251             playerPosition.calculateDistance(nearestPriest)) {
252             nearestPriest = priestPosition;
253         }
254     }
255     //Por cada tile de distancia espera 200ms
256     auto waitingTime = static_cast<double>(playerPosition.calculateDistance(near
estPriest) * 200);
257     player.sendMessage(WAITING_TIME_MESSAGE);
258     player.sendMessage(std::to_string(static_cast<int>(waitingTime/1000)) + " seco
nds\n");
259     playersToResurrect.push_back({waitingTime, 0, nearestPriest, &player});
260     return false;
261 }
262
263 void Game::messagePlayer(Player& playerWhoMessaged, const std::string &playerToM
essage, const std::string &message) {
264     if (players.count(playerToMessage) == 1) {
265         Player* player = players.at(playerToMessage);
266         player->addMessage(message);
267     } else {
268         playerWhoMessaged.addMessage(NO_PLAYER_MESSAGE);
269     }
270 }
271
272 bool Game::playerExists(const std::string &nickname) const {
273     return (players.count(nickname) == 1);
274 }
275
276 void Game::requestRestore(Player& player, Coordinate target) {
277     map.requestRestore(player, target);
278 }
279
280 void Game::getPlayerBank(PlayerData &playerData) {
281     Banker::getPlayerItems(playerData);
282 }
283
284 bool PlayerShouldBeRemoved::operator()(const Player* player) {
285     return (playerToRemove == player);
286 }

```

jul 21, 20 15:47

Withdraw.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_WITHDRAW_H
6  #define ARGENTUM_WITHDRAW_H
7
8
9  #include "Event.h"
10 #include <string>
11 #include "../Map/Coordinate.h"
12
13 class Player;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando withdraw
17 class Withdraw : public Event {
18 private:
19     Player& player;
20     std::string itemName;
21     Coordinate npcPosition{};
22
23 public:
24     Withdraw(Player& player, std::string^ _itemName, Coordinate _npcPosition);
25
26     //Intenta llamar a withdraw en la posicion guardada en el constructor, pidie
ndole
27     //a map que realice esa accion
28     void operator()(ServerProtocol& protocol) override;
29 };
30
31 #endif //ARGENTUM_WITHDRAW_H

```


jul 21, 20 15:47

Withdraw.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Withdraw.h"
6  #include "../Entities/Player.h"
7
8  Withdraw::Withdraw(Player &_player, std::string^ _itemName, Coordinate _npcPosi
tion) : player(_player) {
9      itemName = std::move(_itemName);
10     npcPosition = _npcPosition;
11 }
12
13 void Withdraw::operator()(ServerProtocol& protocol) {
14     player.withdrawFrom(itemName, npcPosition);
15 }

```

jul 21, 20 15:47

UseItem.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_USEITEM_H
6  #define ARGENTUM_USEITEM_H
7
8
9  #include "Event.h"
10
11 class Player;
12
13 //Clase que se almacena en la cola de eventos cuando un player quiere usar
14 //un item
15 class UseItem : public Event {
16 private:
17     Player& player;
18     int position;
19
20 public:
21     UseItem(Player& player, int position);
22
23     //Intenta usar el item del lugar recibido en el constructor, en caso de ser
24     //necesario, le comunica al resto de los clientes si algun (y cual) item fu
e equipado
25     void operator()(ServerProtocol& protocol) override;
26 };
27
28
29 #endif //ARGENTUM_USEITEM_H

```

jul 21, 20 15:47

UseItem.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "UseItem.h"
6  #include "../Entities/Player.h"
7  #include <msgpack.hpp>
8  #include "../Server/ServerProtocol.h"
9
10 MSGPACK_ADD_ENUM(GameType::EventID)
11 MSGPACK_ADD_ENUM(GameType::EquipmentPlace)
12
13 UseItem::UseItem(Player &player, int _position): player(player) {
14     position = _position;
15 }
16
17 void UseItem::operator()(ServerProtocol& protocol) {
18     UseReturnData useData = player.useItem(position);
19     if (useData.equipmentPlace != GameType::EQUIPMENT_PLACE_NONE) {
20         std::stringstream data;
21         msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::EQUIPP
22 ED);
23         msgpack::pack(data, messageTypeData);
24         msgpack::type::tuple<std::string, GameType::EquipmentPlace, int32_t> use
25 DataTuple
26         (player.getNickname(), useData.equipmentPlace, useData.i
27 d);
28         msgpack::pack(data, useDataTuple);
29         protocol.addToGeneralData(data);
30     }
31 }

```

jul 21, 20 15:47

Unequip.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_UNEQUIP_H
6  #define ARGENTUM_UNEQUIP_H
7
8
9  #include "Event.h"
10 #include "../Items/Defense/Clothing.h"
11 #include "../libs/GameEnums.h"
12
13 //Clase que se almacena en la cola de eventos cuando un player quiere desequipar
14 se
15 //un item
16 class Unequip : public Event {
17 private:
18     Player& player;
19     GameType::EquipmentPlace equipment;
20
21 public:
22     Unequip(Player& player, GameType::EquipmentPlace equipment);
23
24     //Intenta desequipar el item del lugar recibido en el constructor, si logra
25 //hacerlo entonces le comunica a los clientes que ese item ya no se encuentr
26 a
27 //equipado, agregandolo al protocolo, si el equipment place es el del pecho,
28 //entonces notifica tambien que el player se equipo ropa default
29 void operator()(ServerProtocol& protocol) override;
30 };
31
32 #endif //ARGENTUM_UNEQUIP_H

```

jul 21, 20 15:47

Unequip.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Unequip.h"
6  #include "../Entities/Player.h"
7  #include "../Items/Item.h"
8  #include "../Server/ServerProtocol.h"
9  #include "msgpack.hpp"
10
11 MSGPACK_ADD_ENUM(GameType::EventID)
12 MSGPACK_ADD_ENUM(GameType::EquipmentPlace)
13
14 Unequip::Unequip(Player &player, GameType::EquipmentPlace _equipment): player(pl
ayer) {
15     equipment = _equipment;
16 }
17
18 void Unequip::operator()(ServerProtocol& protocol) {
19     bool hasAppearanceChanged;
20     if (equipment == GameType::EQUIPMENT_PLACE_WEAPON) {
21         hasAppearanceChanged = player.unequip();
22     } else {
23         hasAppearanceChanged = player.unequip(equipment);
24     }
25     if (hasAppearanceChanged) {
26         std::stringstream data;
27         msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::UNEQUI
P);
28         msgpack::pack(data, messageTypeData);
29         msgpack::type::tuple<std::string, GameType::EquipmentPlace>
30             unequipData(player.getNickname(), equipment);
31         msgpack::pack(data, unequipData);
32         if (equipment == GameType::EQUIPMENT_PLACE_CHEST) {
33             msgpack::type::tuple<GameType::EventID> messageTypeEquipData(GameTyp
e::EQUIPPED);
34             msgpack::pack(data, messageTypeEquipData);
35             msgpack::type::tuple<std::string, GameType::EquipmentPlace, int32_t>
36                 useDataTuple
37                     (player.getNickname(), GameType::EQUIPMENT_PLACE_CHEST, Game
Type::COMMON_CLOTHING);
38             msgpack::pack(data, useDataTuple);
39         }
40         protocol.addToGeneralData(data);
41     }
42 }

```

jul 21, 20 15:47

Sell.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_SELL_H
6  #define ARGENTUM_SELL_H
7
8
9  #include "Event.h"
10 #include "../Map/Coordinate.h"
11
12
13 class Player;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando sell
17 class Sell: public Event {
18 private:
19     Player& player;
20     Coordinate position{};
21     std::string itemName;
22 public:
23     Sell(Player& player, std::string^ itemName, Coordinate position);
24
25     //Intenta llamar a sell en la posicion guardada en el constructor, pidiendol
26     e
27     //a map que realice esa accion
28     void operator()(ServerProtocol& protocol) override;
29 };
30
31 #endif //ARGENTUM_SELL_H

```

jul 21, 20 15:47

Sell.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Sell.h"
6  #include "../Entities/Player.h"
7
8  Sell::Sell(Player &player, std::string^ _itemName, Coordinate _position):
9      player(player), itemName(std::move(_itemName)) {
10      position = _position;
11  }
12
13 void Sell::operator()(ServerProtocol& protocol) {
14     player.sellTo(itemName, position);
15 }

```

jul 21, 20 15:47

RestoreStats.h

Page 1/1

```

1  //
2  // Created by agustin on 10/7/20.
3  //
4
5  #ifndef ARGENTUM_RESTORESTATS_H
6  #define ARGENTUM_RESTORESTATS_H
7
8
9  #include "Event.h"
10 #include "../Map/Coordinate.h"
11
12 class Player;
13 class Game;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando heal
17 class RestoreStats: public Event {
18 private:
19     Game& game;
20     Player& player;
21     Coordinate target;
22
23 public:
24     RestoreStats(Game& game, Player& player, Coordinate target);
25
26     //Pide a game que llame a restore para el mapa en la coordenada recibida
27     void operator()(ServerProtocol& serverProtocol) override;
28 };
29
30
31 #endif //ARGENTUM_RESTORESTATS_H

```

jul 21, 20 15:47

RestoreStats.cpp

Page 1/1

```

1  //
2  // Created by agustin on 10/7/20.
3  //
4
5  #include "RestoreStats.h"
6
7  #include "../Entities/Player.h"
8  #include "../Game.h"
9
10 RestoreStats::RestoreStats(Game &game, Player &player, Coordinate _target): game
    (game), player(player) {
11     target = _target;
12 }
13
14 void RestoreStats::operator()(ServerProtocol &serverProtocol) {
15     game.requestRestore(player, target);
16 }

```

jul 21, 20 15:47

RequestResurrect.h

Page 1/1

```

1  //
2  // Created by agustin on 8/7/20.
3  //
4
5  #ifndef ARGENTUM_REQUESTRESURRECT_H
6  #define ARGENTUM_REQUESTRESURRECT_H
7
8
9  #include "Event.h"
10 #include "../Map/Coordinate.h"
11
12 class Player;
13 class Game;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando resurrect
17 class RequestResurrect: public Event {
18 private:
19     Player& player;
20     Game& game;
21     Coordinate selectedPosition{};
22
23 public:
24     RequestResurrect(Game& game, Player& player, Coordinate selectedPosition);
25
26     //Pide a game que reviva al player, mandandole la coordenada a la que player
27     //le hace el pedido, si es resucitado al realizarlo entonces se agrega esta
28     //informacion al protocolo para que le llegue a todos los clientes y sepan
29     //que el player esta vivo
30     void operator()(ServerProtocol& protocol) override;
31 };
32
33
34 #endif //ARGENTUM_REQUESTRESURRECT_H

```

jul 21, 20 15:47

RequestResurrect.cpp

Page 1/1

```

1  //
2  // Created by agustin on 8/7/20.
3  //
4
5  #include "RequestResurrect.h"
6
7  #include "../Entities/Player.h"
8  #include "../Game.h"
9  #include "../Server/ServerProtocol.h"
10 #include <msgpack.hpp>
11 #include "../Server/ServerProtocol.h"
12
13 MSGPACK_ADD_ENUM(GameType::EventID)
14
15 RequestResurrect::RequestResurrect(Game& game, Player& player, Coordinate _selectedPosition)
16                                     : player(player), game(game) {
17     selectedPosition = _selectedPosition;
18 }
19
20 void RequestResurrect::operator()(ServerProtocol &protocol) {
21     if (game.requestResurrect(player, selectedPosition)) {
22         std::stringstream data;
23         msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::RESURRECTED);
24         msgpack::pack(data, messageTypeData);
25         msgpack::type::tuple<std::string> playerData(player.getNickname());
26         msgpack::pack(data, playerData);
27         protocol.addToGeneralData(data);
28     }
29 }

```

jul 21, 20 15:47

PlayerLeveledUp.h

Page 1/1

```

1  //
2  // Created by agustin on 20/7/20.
3  //
4
5  #ifndef ARGENTUM_PLAYERLEVELEDUP_H
6  #define ARGENTUM_PLAYERLEVELEDUP_H
7
8
9  #include "Event.h"
10
11 #include <string>
12
13 class PlayerLeveledUp: public Event {
14 private:
15     const std::string& playerNickname;
16     int32_t level;
17
18 public:
19     explicit PlayerLeveledUp(const std::string& playerNickname, int32_t level);
20
21     void operator()(ServerProtocol& protocol) override;
22 };
23
24
25 #endif //ARGENTUM_PLAYERLEVELEDUP_H

```

jul 21, 20 15:47

PlayerLeveledUp.cpp

Page 1/1

```

1  //
2  // Created by agustin on 20/7/20.
3  //
4
5  #include "PlayerLeveledUp.h"
6
7  #include "../libs/GameEnums.h"
8  #include "../Server/ServerProtocol.h"
9  #include <msgpack.hpp>
10
11 MSGPACK_ADD_ENUM(GameType::EventID)
12
13 PlayerLeveledUp::PlayerLeveledUp(const std::string &playerNickname, int32_t _level):
14     playerNickname(playerNickname) {
15     level = _level;
16 }
17
18 void PlayerLeveledUp::operator()(ServerProtocol &protocol) {
19     std::stringstream data;
20     msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::PLAYER_LEVEL_UP);
21     msgpack::pack(data, messageTypeData);
22     msgpack::type::tuple<std::string, int32_t> nicknameData(playerNickname, level);
23     msgpack::pack(data, nicknameData);
24     protocol.addToGeneralData(data);
25 }

```

jul 21, 20 15:47

PickUpItem.h

Page 1/1

```

1  //
2  // Created by agustin on 6/7/20.
3  //
4
5  #ifndef ARGENTUM_PICKUPITEM_H
6  #define ARGENTUM_PICKUPITEM_H
7
8
9  #include "Event.h"
10
11 class Game;
12 class Player;
13 class ServerProtocol;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando pickUp
17 class PickUpItem: public Event {
18 private:
19     Game& game;
20     Player& player;
21
22 public:
23     PickUpItem(Game& game, Player& player);
24
25     //Le dice a game que el player quiere agarrar el item que se encuentra en
26     //su posicion, si lo logra entonces manda al cliente el mensaje necesario:
27     //Manda que se debe destruir el item de esa posicion si es que ya no quedan
28     //mas items, sino manda que se debe crear el item que se quiere mostrar en
29     //esa posicion
30     void operator()(ServerProtocol& protocol) override;
31 };
32
33
34 #endif //ARGENTUM_PICKUPITEM_H

```

jul 21, 20 15:47

PickUpItem.cpp

Page 1/1

```

1  //
2  // Created by agustin on 6/7/20.
3  //
4
5  #include "PickUpItem.h"
6  #include "../Entities/Player.h"
7  #include "../Game.h"
8  #include "../Server/ServerProtocol.h"
9  #include <msgpack.hpp>
10
11 MSGPACK_ADD_ENUM(GameType::EventID)
12 MSGPACK_ADD_ENUM(GameType::ItemType)
13
14 PickUpItem::PickUpItem(Game &game, Player &player): game(game), player(player) {
15
16 }
17
18 void PickUpItem::operator()(ServerProtocol& protocol) {
19     const Item* itemPtr = game.storeItemFromTileInPlayer(player);
20     std::stringstream data;
21     Coordinate pickUpPosition = player.getPosition();
22     if (itemPtr) {
23         itemPtr->loadDropItemData(data, pickUpPosition.iPosition, pickUpPosition
24         .jPosition);
25         protocol.addToGeneralData(data);
26     } else {
27         msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::DESTRO
28         Y_ITEM);
29         msgpack::pack(data, messageTypeData);
30         msgpack::type::tuple<int32_t, int32_t> itemDataTuple
31         (pickUpPosition.iPosition, pickUpPosition.jPosition);
32         msgpack::pack(data, itemDataTuple);
33         protocol.addToGeneralData(data);
34     }
35 }

```

jul 21, 20 15:47

NotifyDeath.h

Page 1/1

```

1  //
2  // Created by agustin on 5/7/20.
3  //
4
5  #ifndef ARGENTUM_NOTIFYDEATH_H
6  #define ARGENTUM_NOTIFYDEATH_H
7
8
9  #include "Event.h"
10
11 #include <sstream>
12
13 class Player;
14
15 //Clase que se almacena en la cola de eventos para notificarle a los clientes qu
16 e
17 //un player murio, por lo que ahora es un fantasma
18 class NotifyDeath: public Event {
19 private:
20     const Player& player;
21
22 private:
23     void _appendUnequipMessages(std::stringstream& data);
24
25 public:
26     explicit NotifyDeath(const Player& player);
27
28     //Guarda en el protocolo los mensajes de muerte y desequipamiento de items,
29     //ademas del equipamiento de los items default, para que se envíe a todos lo
30 s
31     //clientes
32     void operator()(ServerProtocol& protocol) override;
33 };
34
35 #endif //ARGENTUM_NOTIFYDEATH_H

```


jul 21, 20 15:47

NotifyDeath.cpp

Page 1/1

```

1  //
2  // Created by agustin on 5/7/20.
3  //
4
5  #include "NotifyDeath.h"
6
7  #include "../Entities/Player.h"
8  #include "../libs/GameEnums.h"
9  #include "../Server/ServerProtocol.h"
10 #include <msgpack.hpp>
11
12 MSGPACK_ADD_ENUM(GameType::EventID)
13 MSGPACK_ADD_ENUM(GameType::EquipmentPlace)
14
15 NotifyDeath::NotifyDeath(const Player &player): player(player) {
16 }
17
18
19 void NotifyDeath::operator()(ServerProtocol &protocol) {
20     std::stringstream data;
21     msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::PLAYER_DEATH);
22     msgpack::pack(data, messageTypeData);
23     msgpack::type::tuple<std::string> nicknameData(player.getNickname());
24     msgpack::pack(data, nicknameData);
25     _appendUnequipMessages(data);
26     protocol.addToGeneralData(data);
27 }
28
29 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
30
31 void NotifyDeath::_appendUnequipMessages(std::stringstream& data) {
32     std::vector<GameType::EquipmentPlace> equipment = {GameType::EQUIPMENT_PLACE_WEAPON,
33     GameType::EQUIPMENT_PLACE_SHIELD, GameType::EQUIPMENT_PLACE_HEAD,
34     GameType::EQUIPMENT_PLACE_CHEST};
35
36     for (const auto & place: equipment) {
37         msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::UNEQUIPPED);
38         msgpack::pack(data, messageTypeData);
39         msgpack::type::tuple<std::string, GameType::EquipmentPlace> unequipData(player.getNickname(), place);
40         msgpack::pack(data, unequipData);
41     }
42     msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::EQUIPPED);
43     msgpack::pack(data, messageTypeData);
44     msgpack::type::tuple<std::string, GameType::EquipmentPlace, int32_t> equippedData(player.getNickname(), GameType::EQUIPMENT_PLACE_CHEST,
45     GameType::COMMON_CLOTHING);
46     msgpack::pack(data, equippedData);
47 }
48
49 }

```

jul 21, 20 15:47

Move.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_MOVE_H
6  #define ARGENTUM_MOVE_H
7
8
9  #include "Event.h"
10 #include "../Map/Coordinate.h"
11 #include "../libs/GameEnums.h"
12
13 class Game;
14 class Entity;
15
16 //Clase que se almacena en la cola de eventos cuando un entity quiere concretar su
17 //desplazamiento en alguna direccion
18 class Move: public Event {
19 private:
20     Game& game;
21     Entity& entity;
22     GameType::Direction moveDirection;
23
24 public:
25     Move(Game& _game, Entity& _entity, GameType::Direction moveDirection);
26
27     //Intenta mover el entity guardado en la direccion recibida en el constructor,
28     //pasandola inmediatamente al tile si es que esta disponible para el entity,
29     //empezando asi su desplazamiento "visual" hacia este
30     void operator()(ServerProtocol& protocol) override;
31 };
32
33
34 #endif //ARGENTUM_MOVE_H

```

jul 21, 20 15:47

Moved.h

Page 1/1

```

1  //
2  // Created by agustin on 29/6/20.
3  //
4
5  #ifndef ARGENTUM_MOVED_H
6  #define ARGENTUM_MOVED_H
7
8
9  #include "Event.h"
10 #include "../libs/GameEnums.h"
11
12 #include <cstdint>
13
14 class Entity;
15 class ServerProtocol;
16
17 //Clase que se almacena en la cola de eventos para notificarle a los clientes qu
18 e
19 //un entity se desplazo
20 class Moved: public Event {
21 private:
22     Entity& entity;
23     GameType::Direction direction;
24     int32_t displacement;
25
26 public:
27     Moved(Entity& entity, GameType::Direction direction, int32_t displacement);
28
29     //Almacena en el protocolo el mensaje del desplazamiento de un entity para
30     //comunicarselo a todos los clientes
31     void operator()(ServerProtocol& protocol) override;
32 };
33
34 #endif //ARGENTUM_MOVED_H

```

jul 21, 20 15:47

Moved.cpp

Page 1/1

```

1  //
2  // Created by agustin on 29/6/20.
3  //
4
5  #include "Moved.h"
6  #include "../Entities/Entity.h"
7  #include "../Server/ServerProtocol.h"
8  #include <msgpack.hpp>
9
10 MSGPACK_ADD_ENUM(GameType::EventID)
11 MSGPACK_ADD_ENUM(GameType::Direction)
12
13 Moved::Moved(Entity &entity, GameType::Direction _direction, int32_t _displaceme
14 nt):
15     entity(entity) {
16     direction = _direction;
17     displacement = _displacement;
18 }
19
20 void Moved::operator()(ServerProtocol &protocol) {
21     std::stringstream data;
22     msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::MOVED);
23     msgpack::pack(data, messageTypeData);
24     bool hasFinishedMoving;
25     int32_t realDisplacement = entity.executeDisplacement(displacement, hasFinis
26 hedMoving);
27     msgpack::type::tuple<GameType::Direction, int32_t, std::string, bool>
28     eventData(direction, realDisplacement, entity.getNicknam
29 e(), hasFinishedMoving);
30     msgpack::pack(data, eventData);
31     protocol.addToGeneralData(data);
32 }

```

jul 21, 20 15:47

Move.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Move.h"
6  #include "../Entities/Entity.h"
7  #include "../Game.h"
8
9  Move::Move(Game &_game, Entity &_entity, GameType::Direction _moveDirection) :
10                                     game(_game), entity(_entity) {
11      moveDirection = _moveDirection;
12  }
13
14  void Move::operator()(ServerProtocol& protocol) {
15      game.moveEntity(entity.getPosition(), entity.getFinalCoordinate(moveDirectio
16      n));
17  }

```

jul 21, 20 15:47

ModifyPlayerMovement.h

Page 1/1

```

1  //
2  // Created by agustin on 14/7/20.
3  //
4
5  #ifndef ARGENTUM_MODIFYPLAYERMOVEMENT_H
6  #define ARGENTUM_MODIFYPLAYERMOVEMENT_H
7
8
9  #include "Event.h"
10 #include "../libs/GameEnums.h"
11
12 class Player;
13
14 //Clase que se almacena en la cola de eventos cuando un player quiere desplazars
15 e
16 //en alguna direccion
17 class ModifyPlayerMovement: public Event {
18 private:
19     Player& player;
20     GameType::Direction direction{};
21     bool continuesMovement;
22 public:
23     //Constructor que le indica a movement que se debe empezar a mover en la dir
24     eccion
25     //recibida
26     explicit ModifyPlayerMovement(Player& player, GameType::Direction direction)
27     ;
28
29     //constructor que le indica a movement que debe dejar de moverse
30     explicit ModifyPlayerMovement(Player& player);
31
32     //Guarda el proximo estado de movimiento para que sea ejecutado una vez que
33     //termine de realizar el movimiento que se este realizando en el momento
34     void operator()(ServerProtocol& protocol) override;
35 };
36
37 #endif //ARGENTUM_MODIFYPLAYERMOVEMENT_H

```

jul 21, 20 15:47

ModifyPlayerMovement.cpp

Page 1/1

```

1  //
2  // Created by agustin on 14/7/20.
3  //
4
5  #include "ModifyPlayerMovement.h"
6
7  #include "../Entities/Player.h"
8
9  ModifyPlayerMovement::ModifyPlayerMovement(Player& player, GameType::Direction _
direction): player(player) {
10     direction = _direction;
11     continuesMovement = true;
12 }
13
14 ModifyPlayerMovement::ModifyPlayerMovement(Player& player): player(player) {
15     continuesMovement = false;
16 }
17
18 void ModifyPlayerMovement::operator()(ServerProtocol& protocol) {
19     if (continuesMovement) {
20         player.startMovement(direction);
21     } else {
22         player.stopMovement();
23     }
24 }

```

jul 21, 20 15:47

Message.h

Page 1/1

```

1  //
2  // Created by marcos on 7/10/20.
3  //
4
5  #ifndef ARGENTUM_MESSAGE_H
6  #define ARGENTUM_MESSAGE_H
7
8  #include "Event.h"
9  #include <string>
10
11 class Game;
12 class Player;
13
14 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
15 //el comando message
16 class Message : public Event {
17 private:
18     Game& game;
19     Player& playerWhoMessaged;
20     std::string playerToMessage;
21     std::string message;
22
23 public:
24     Message(Game& _game, Player& _playerWhoMessaged, std::string^ _playerToMess
age,
25             std::string^ _message);
26
27     //Llama a la funcion de game que manda el mensaje a otro jugador, agregandos
elo
28     //a su minichat
29     void operator()(ServerProtocol& protocol) override;
30 };
31
32
33 #endif //ARGENTUM_MESSAGE_H

```

jul 21, 20 15:47

Message.cpp

Page 1/1

```

1  //
2  // Created by marcos on 7/10/20.
3  //
4
5  #include "Message.h"
6  #include "../Entities/Player.h"
7  #include "../Game.h"
8
9  void Message::operator()(ServerProtocol &protocol) {
10     game.messagePlayer(playerWhoMessaged, playerToMessage, message);
11 }
12
13 Message::Message(Game &_game, Player &_playerWhoMessaged,
14                 std::string ^_playerToMessage, std::string ^_message)
15     : game(_game), playerWhoMessaged(_playerWhoMessaged), playerToM
16     essage(std::move(_playerToMessage)) {
17     message = _playerWhoMessaged.getNickname() + ":" + _message;
18 }

```

jul 21, 20 15:47

List.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_LIST_H
6  #define ARGENTUM_LIST_H
7
8
9  #include "Event.h"
10 #include <string>
11 #include "../Map/Coordinate.h"
12
13 class Player;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando list
17 class List : public Event {
18 private:
19     Player& player;
20     Coordinate npcPosition{};
21
22 public:
23     List(Player& player, Coordinate _npcPosition);
24
25     //Intenta llamar a list en la posicion guardada en el constructor, pidiendol
26     e
27     //a map que realice esa accion
28     void operator()(ServerProtocol& protocol) override;
29 };
30
31 #endif //ARGENTUM_LIST_H

```

jul 21, 20 15:47

List.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "List.h"
6  #include "../Entities/Player.h"
7
8  List::List(Player &_player, Coordinate _npcPosition) : player(_player) {
9      npcPosition = _npcPosition;
10 }
11
12 void List::operator()(ServerProtocol& protocol) {
13     player.listFrom(npcPosition);
14 }

```

jul 21, 20 15:47

GetInventoryNames.h

Page 1/1

```

1  //
2  // Created by agustin on 13/7/20.
3  //
4
5  #ifndef ARGENTUM_GETINVENTORYNAMES_H
6  #define ARGENTUM_GETINVENTORYNAMES_H
7
8
9  #include "Event.h"
10
11 class Player;
12
13 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
14 //el comando inventory
15 class GetInventoryNames: public Event {
16 private:
17     Player& player;
18
19 public:
20     explicit GetInventoryNames(Player& player);
21
22     //Guarda en el minichat del player almacenado mensajes que indican el nombre
23     //de cada item del inventario junto a la posicion del inventario en el que
24     //este se encuentra. De la forma posicion: Nombre
25     void operator()(ServerProtocol& protocol) override;
26 };
27
28
29 #endif //ARGENTUM_GETINVENTORYNAMES_H

```

jul 21, 20 15:47

GetInventoryNames.cpp

Page 1/1

```

1  //
2  // Created by agustin on 13/7/20.
3  //
4
5  #include "GetInventoryNames.h"
6
7  #include "../Entities/Player.h"
8
9  GetInventoryNames::GetInventoryNames(Player &player): player(player) {
10
11 }
12
13 void GetInventoryNames::operator()(ServerProtocol &protocol) {
14     player.getInventoryNames();
15 }

```

jul 21, 20 15:47

Event.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_EVENT_H
6  #define ARGENTUM_EVENT_H
7
8  class ServerProtocol;
9
10 //Interfaz para encolar eventos para game
11 class Event {
12 public:
13
14     //Funcion a implementar para cada clase que herede de esta, debe realizar
15     //la accion del juegoq que se quiere ejecutar
16     virtual void operator()(ServerProtocol& protocol) = 0;
17     virtual ~Event() = default;
18 };
19
20
21 #endif //ARGENTUM_EVENT_H

```

jul 21, 20 15:47

Event.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Event.h"

```

jul 21, 20 15:47

Drop.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_DROP_H
6  #define ARGENTUM_DROP_H
7
8
9  #include "Event.h"
10
11 #include <string>
12 #include <memory>
13 #include <list>
14 #include "../Items/Item.h"
15 #include "../Map/Coordinate.h"
16
17 class ServerProtocol;
18 class Player;
19 class Game;
20
21 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
22 //el comando drop
23 class Drop: public Event {
24 private:
25     Player* player{nullptr};
26     int inventoryPosition{-1};
27
28     std::list<std::shared_ptr<Item>> items;
29     Game* game{nullptr};
30     Coordinate dropPosition{-1, -1};
31
32 public:
33     Drop(Player& player, int position);
34
35     //La lista debe contener al menos 1 elemento si se utiliza este constructor,
36     //sino
37     //se tirara la excepcion std::invalid_argument
38     Drop(Game& game, std::list<std::shared_ptr<Item>>& items, Coordinate dropPosition);
39     Drop(Game& game, std::shared_ptr<Item>& item, Coordinate dropPosition);
40
41     //Si se llamo al primer constructor entonces se intenta sacar el item de la
42     //posicion
43     //del inventario guardada y, si hay un item en esta, se envia al cliente que
44     //item debe mostrar
45     //el item en el piso, ademas de guardarlo en el tile en el que se encuentra
46     //parado el player
47     //Si se llamo al segundo o tercer constructor, se agregan todos los items recibidos
48     //en el tile en el que se desean dejar y se manda al cliente el item que debe mostrar
49     //en ese tile
50     void operator()(ServerProtocol& protocol) override;
51 };
52
53 #endif //ARGENTUM_DROP_H

```


jul 21, 20 15:47

Drop.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Drop.h"
6  #include "../Entities/Player.h"
7  #include "../Items/ItemData.h"
8  #include "../Server/ServerProtocol.h"
9  #include "../Game.h"
10 #include "../Items/Item.h"
11 #include <msgpack.hpp>
12
13 MSGPACK_ADD_ENUM(GameType::EventID)
14 MSGPACK_ADD_ENUM(GameType::ItemType)
15
16
17 Drop::Drop(Player &_player, int _inventoryPosition) {
18     player = &_player;
19     inventoryPosition = _inventoryPosition;
20     dropPosition = player->getPosition();
21 }
22
23 Drop::Drop(Game& _game, std::list<std::shared_ptr<Item>>& _items, Coordinate_d
ropPosition) {
24     game = &_game;
25     dropPosition = _dropPosition;
26     if (_items.empty()) {
27         throw std::invalid_argument("List without elements in Drop");
28     }
29     items = std::move(_items);
30 }
31
32 Drop::Drop(Game &_game, std::shared_ptr<Item> ^item, Coordinate _dropPosition)
{
33     game = &_game;
34     dropPosition = _dropPosition;
35     items.push_back(std::move(item));
36 }
37
38 void Drop::operator()(ServerProtocol& protocol) {
39     const Item* itemPtr = nullptr;
40     if (player) {
41         itemPtr = player->dropItem(inventoryPosition);
42     } else {
43         itemPtr = items.back().get();
44         game->dropItems(std::move(items), dropPosition);
45     }
46     if (itemPtr) {
47         std::stringstream data;
48         itemPtr->loadDropItemData(data, dropPosition.iPosition, dropPosition.jPo
sition);
49         protocol.addToGeneralData(data);
50     }
51 }

```

jul 21, 20 15:47

Deposit.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_DEPOSIT_H
6  #define ARGENTUM_DEPOSIT_H
7
8
9  #include "Event.h"
10 #include <string>
11 #include "../Map/Coordinate.h"
12
13 class Player;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando de deposit en un tile
17 class Deposit : public Event {
18 private:
19     Player& player;
20     std::string itemName;
21     Coordinate npcPosition{};
22
23 public:
24     Deposit(Player& player, std::string^ _itemName, Coordinate _npcPosition);
25
26     //Intenta depositar el item con el nombre en el entity que se encuentre
27     //en la coordenada guardada, pidiendole a map que realice esa accion
28     void operator()(ServerProtocol& protocol) override;
29
30 };
31
32
33 #endif //ARGENTUM_DEPOSIT_H

```

jul 21, 20 15:47

Deposit.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Deposit.h"
6  #include "../Entities/Player.h"
7
8  Deposit::Deposit(Player &player, std::string^ _itemName, Coordinate _npcPositi
on) : player(_player) {
9      itemName = std::move(_itemName);
10     npcPosition = _npcPosition;
11 }
12
13 void Deposit::operator()(ServerProtocol& protocol) {
14     player.depositTo(itemName, npcPosition);
15 }

```

jul 21, 20 15:47

Buy.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_BUY_H
6  #define ARGENTUM_BUY_H
7
8
9  #include "Event.h"
10 #include <string>
11 #include "../Map/Coordinate.h"
12
13 class Player;
14
15 //Clase que se almacena en la cola de eventos cuando un player quiere ejecutar
16 //el comando de comprar en un tile
17 class Buy : public Event {
18 private:
19     Player& player;
20     std::string itemName;
21     Coordinate npcPosition{};
22
23 public:
24     Buy(Player& player, std::string^ _itemName, Coordinate _npcPosition);
25
26     //Intenta comprar el item con el nombre guardado al entity que se encuentre
27     //en la coordenada guardada, pidiendole a map que realice esa accion
28     void operator()(ServerProtocol& protocol) override;
29 };
30
31
32 #endif //ARGENTUM_BUY_H

```

jul 21, 20 15:47

Buy.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Buy.h"
6  #include "../Entities/Player.h"
7
8  Buy::Buy(Player &_player, std::string^ _itemName, Coordinate _npcPosition) : pl
ayer(_player) {
9      itemName = std::move(_itemName);
10     npcPosition = _npcPosition;
11 }
12
13 void Buy::operator()(ServerProtocol& protocol) {
14     player.buyFrom(itemName, npcPosition);
15 }

```

jul 21, 20 15:47

Attack.h

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #ifndef ARGENTUM_ATTACK_H
6  #define ARGENTUM_ATTACK_H
7
8
9  #include "Event.h"
10 #include "../Map/Coordinate.h"
11 #include "../libs/GameEnums.h"
12
13 class Entity;
14
15 //Clase que se almacena en la cola de eventos cuando se quiere que un entity
16 //ataque a un tile
17 class Attack : public Event {
18 private:
19     Entity& entity;
20     Coordinate target{};
21
22 public:
23     Attack(Entity& _entity, Coordinate _target);
24
25     //Realiza el ataque y de ser necesario almacena el evento en el protocolo
26     //para comunicarselo a los clientes, pidiendole a map que realice esa accion
27     void operator()(ServerProtocol& protocol) override;
28
29 private:
30     GameType::Direction _attackDirection(Coordinate attackerPosition) const;
31 };
32
33
34 #endif //ARGENTUM_ATTACK_H

```

jul 21, 20 15:47

Attack.cpp

Page 1/1

```

1  //
2  // Created by agustin on 23/6/20.
3  //
4
5  #include "Attack.h"
6  #include "../Entities/Entity.h"
7  #include "../Server/ServerProtocol.h"
8  #include "msgpack.hpp"
9
10 MSGPACK_ADD_ENUM(GameType::Direction)
11 MSGPACK_ADD_ENUM(GameType::EventID)
12
13 Attack::Attack(Entity& _entity, Coordinate _target) : entity(_entity) {
14     target = _target;
15 }
16
17 void Attack::operator()(ServerProtocol& protocol) {
18     int32_t usedWeapon = entity.attack(target);
19     if (usedWeapon != -1) {
20         GameType::Direction attackDir = _attackDirection(entity.getPosition());
21         std::stringstream data;
22         msgpack::type::tuple<GameType::EventID> messageTypeData(GameType::ATTACK
23     );
24         msgpack::pack(data, messageTypeData);
25         msgpack::type::tuple<std::string, int32_t, int32_t, int32_t,
26             GameType::Direction> attackCoordinateData
27             (entity.getNickname(), target.iPosition, target.
28             jPosition,
29             usedWeapon, attackDir);
30         msgpack::pack(data, attackCoordinateData);
31         protocol.addToGeneralData(data);
32     }
33 }
34
35 GameType::Direction Attack::_attackDirection(Coordinate attackerPosition) const
36 {
37     if (attackerPosition.iPosition == target.iPosition) {
38         if (attackerPosition.jPosition > target.jPosition) {
39             return GameType::DIRECTION_LEFT;
40         } else {
41             return GameType::DIRECTION_RIGHT;
42         }
43     } else {
44         if (attackerPosition.iPosition < target.iPosition) {
45             return GameType::DIRECTION_DOWN;
46         } else {
47             return GameType::DIRECTION_UP;
48         }
49     }
50 }

```

jul 21, 20 15:47

UnavailablePlayerException.h

Page 1/1

```

1  //
2  // Created by marcos on 17/7/20.
3  //
4
5  #ifndef ARGENTUM_UNAVAILABLEPLAYEREXCEPTION_H
6  #define ARGENTUM_UNAVAILABLEPLAYEREXCEPTION_H
7
8  #include <exception>
9
10 class UnavailablePlayerException : std::exception {
11
12 };
13
14
15 #endif //ARGENTUM_UNAVAILABLEPLAYEREXCEPTION_H

```

jul 21, 20 15:47

InexistentPlayerException.h

Page 1/1

```

1  //
2  // Created by marcos on 17/7/20.
3  //
4
5  #ifndef ARGENTUM_INEXISTENTPLAYEREXCEPTION_H
6  #define ARGENTUM_INEXISTENTPLAYEREXCEPTION_H
7
8  #include <exception>
9
10 class InexistentPlayerException : std::exception {
11
12 };
13
14
15 #endif //ARGENTUM_INEXISTENTPLAYEREXCEPTION_H

```

jul 21, 20 15:47

PlayerStats.h

Page 1/2

```

1  //
2  // Created by agustin on 14/6/20.
3  //
4
5  #ifndef ARGENTUM_PLAYERSTATS_H
6  #define ARGENTUM_PLAYERSTATS_H
7
8  #include <sstream>
9  #include <stdint>
10 #include "../libs/GameEnums.h"
11 #include "../Server/PlayerData.hpp"
12 #include "../Config/ConfigFileReader.h"
13 #include "../Config/Configuration.h"
14
15 class EntityTests;
16 class MapTests;
17 class Minichat;
18
19 class PlayerStats {
20 private:
21     bool isMeditating{};
22     double timeElapsedLife{};
23     double timeElapsedMana{};
24
25     int32_t constitution{};
26     int32_t intelligence{};
27     int32_t agility{};
28     int32_t strength{};
29
30     int32_t classLifeMultiplier{};
31     int32_t raceLifeMultiplier{};
32     int32_t classManaMultiplier{};
33     int32_t raceManaMultiplier{};
34     int32_t recoveryRate{};
35     int32_t meditationRate{};
36
37     int32_t experience{};
38     int32_t nextLevelExperience{};
39     int32_t level{};
40     int32_t currentMana{};
41     int32_t currentLife{};
42     int32_t maxMana{};
43     int32_t maxLife{};
44
45     const double TIME_FOR_RECOVERY{Configuration::getInstance().configPlayerRecoveryTime()*1000};
46
47     friend EntityTests;
48     friend MapTests;
49
50 public:
51     //Construye el PlayerStats utilizando los datos almacenados en la instancia de PlayerData
52     //Esta pensado para cargar los stats que tuvo un jugador antes de desconectar
53     explicit PlayerStats(const PlayerData& data);
54
55     //Retorna el danio base que logro hacer el arma del player para el ataque
56     int getTotalDamage(int weaponDamage) const;
57
58     //Retorna el level actual del player
59     unsigned int getLevel() const;
60
61     //Aumenta la xp del player, retorna true si subio de nivel, false en otro caso
62     //Un player nunca puede subir de a mas de un nivel ya que la experiencia de

```

jul 21, 20 15:47

PlayerStats.h

Page 2/2

```

63 //sobra es descartada
64 bool increaseExperience(unsigned int _experience);
65
66 //Modifica la vida del player acorde al dano/curacion ocasionados
67 //Retorna el pair(danio total recibido, pudo esquivar)
68 //Concatena en attackedMessage prefijos para el mensaje de ataque segun el
69 //resultado
70 std::pair<int, bool> modifyLife(int damage, unsigned int attackerLevel, unsigned int defense,
71                                bool isAPlayer, std::string& attackedMessage);
72
73 //Retorna la maxima vida que puede tener el player dados sus stats actuales
74 int getMaxLife() const;
75
76 //Retorna la vida actual del player
77 int getCurrentLife() const;
78
79 //Restaura hasta amount cantidad de vida, sin pasarse de la cantidad maxima
80 void restoreLife(unsigned int amount);
81
82 //Restaura hasta amount cantidad de mana, sin pasarse de la cantidad maxima
83 void restoreMana(unsigned int amount);
84
85 //Retorna si el player esta o no muerto
86 bool isDead() const;
87
88 //Actualiza el estado de la vida y el mana del player
89 void update(double timeStep);
90
91 //Setea el player a modo meditacion
92 void startMeditating(Minichat& minichat);
93
94 //Hace que el player deje de estar en modo meditacion
95 void stopMeditating(Minichat& minichat);
96
97 int32_t& getCurrentMana();
98
99 //Setea el mana y la vida actual en sus valores maximos
100 void restore();
101
102 //Guarda experiencia, experiencia para proximo nivel, nivel, mana actual, mana maximo,
103 //vida, vida maxima y si esta vivo (true) o muerto (false) en buffer
104 void storeAllRelevantData(std::stringstream& buffer) const;
105
106 //Guarda true si esta vivo, sino guarda false en buffer
107 void storeLifeStatus(std::stringstream& buffer) const;
108
109 //Almacena las stats del player en pData, se usa para el backup del archivo
110 void getData(PlayerData& pData) const;
111
112 //Intenta consumir amount cantidad de mana, retorna true si lo pudo hacer,
113 //sino retorna false
114 bool consumeMana(unsigned int amount);
115
116 private:
117 void _increaseStats();
118 void _loadInitialStats(Config::Modifiers& classM, Config::Modifiers& raceM,
119                        const PlayerData& data);
120 void _loadGenericStats(Config::Modifiers& classM, Config::Modifiers& raceM,
121                        const PlayerData& data);
122 };
123
124 #endif //ARGENTUM_PLAYERSTATS_H

```

jul 21, 20 15:47

PlayerStats.cpp

Page 1/5

```

1 //
2 // Created by agustin on 14/6/20.
3 //
4
5 #include "PlayerStats.h"
6 #include "../Config/Calculator.h"
7 #include "Minichat.h"
8 #include <algorithm>
9 #include <msgpack.hpp>
10
11 #define MUCH_LEVEL_DIFF_MESSAGE "I think the level gap between us is a tad much, I'm level "
12 #define CRITICAL_MESSAGE "That must have hurt! Critical! "
13 #define DODGE_MESSAGE "Too weak, too slow. "
14 #define STARTED_MEDITATING_MESSAGE "Started meditating\n"
15 #define STOPPED_MEDITATING_MESSAGE "Stopped meditating\n"
16
17 using namespace GameType;
18
19 //////////////////////////////////////////PUBLIC////////////////////////////////////////
20
21 //Funcion auxiliar para inicializar el resto de los atributos de la clase
22 void PlayerStats::_loadGenericStats(Config::Modifiers& classM, Config::Modifiers& raceM,
23                                     const PlayerData& data) {
24     isMeditating = false;
25     timeElapsedLife = 0;
26     timeElapsedMana = 0;
27     experience = data.experience;
28     level = data.level;
29     maxLife = Calculator::calculateMaxLife(constitution, classLifeMultiplier, raceLifeMultiplier, level);
30     maxMana = Calculator::calculateMaxMana(intelligence, classManaMultiplier, raceManaMultiplier, level);
31     currentLife = maxLife;
32     currentMana = maxMana;
33     nextLevelExperience = Calculator::calculateNextLevelXP(level);
34     recoveryRate = raceM.recoveryRate;
35     meditationRate = classM.meditationRate;
36 }
37
38 //Setea los valores de los stats del player segun sea un jugador nuevo o uno ya existente
39 void PlayerStats::_loadInitialStats(Config::Modifiers& classM, Config::Modifiers& raceM,
40                                     const PlayerData& data) {
41     if (data.isNewPlayer) {
42         constitution += classM.constitution + raceM.constitution;
43         intelligence += classM.intelligence + raceM.intelligence;
44         agility += classM.agility + raceM.agility;
45         strength += classM.strength + raceM.strength;
46     } else {
47         constitution = data.constitution;
48         intelligence = data.intelligence;
49         agility = data.agility;
50         strength = data.strength;
51     }
52 }
53
54 //Aumenta en 1 el nivel de los atributos guardados y recalcula los nuevos valores
55 //maximos de mana y vida, asignandole tambien estos a los valores actuales
56 void PlayerStats::_increaseStats() {
57     ++strength;
58     ++agility;
59     ++intelligence;
60     ++constitution;

```

jul 21, 20 15:47

PlayerStats.cpp

Page 2/5

```

61     maxLife = Calculator::calculateMaxLife(constitution, classLifeMultiplier, ra
        ceLifeMultiplier,
62                                     level);
63     maxMana = Calculator::calculateMaxMana(intelligence, classManaMultiplier, ra
        ceManaMultiplier,
64                                     level);
65     currentLife = maxLife;
66     currentMana = maxMana;
67 }
68
69 ///////////////////////////////////////////////////PUBLIC////////////////////////////////////
70
71 PlayerStats::PlayerStats(const PlayerData& data) {
72     Configuration& config = Configuration::getInstance();
73     Config::Modifiers classModifier = config.configClassModifiers(data.pClass);
74     Config::Modifiers raceModifier = config.configRaceModifiers(data.pRace);
75     classLifeMultiplier = classModifier.lifeMultiplier;
76     classManaMultiplier = classModifier.manaMultiplier;
77     raceLifeMultiplier = raceModifier.lifeMultiplier;
78     raceManaMultiplier = raceModifier.manaMultiplier;
79     _loadInitialStats(classModifier, raceModifier, data);
80     _loadGenericStats(classModifier, raceModifier, data);
81 }
82
83 int PlayerStats::getTotalDamage(int weaponDamage) const {
84     return Calculator::calculateDamage(strength, weaponDamage);
85 }
86
87 unsigned int PlayerStats::getLevel() const {
88     return level;
89 }
90
91 bool PlayerStats::increaseExperience(unsigned int _experience) {
92     experience += _experience;
93     if (experience ≥ nextLevelExperience) {
94         ++level;
95         _increaseStats();
96         experience = 0;
97         nextLevelExperience = Calculator::calculateNextLevelXP(level);
98         return true;
99     }
100     return false;
101 }
102
103 std::pair<int, bool> PlayerStats::modifyLife(int damage, unsigned int attackerLe
        vel, unsigned int defense,
104                                     bool isAPlayer, std::string& attackedMessage) {
105     if (damage < 0) {
106         currentLife += -damage;
107         if (currentLife > maxLife) currentLife = maxLife;
108         currentMana = 0;
109         return {damage, false};
110     } else {
111         Configuration& config = Configuration::getInstance();
112         if (isAPlayer ^ std::abs(static_cast<int32_t>(attackerLevel) - level) >
113             static_cast<int32_t>(config.configMaxLevelDi
114 f())) {
115             attackedMessage += MUCH_LEVEL_DIFF_MESSAGE + std::to_string(level) +
116                 "\n";
117             return {0, false};
118         }
119         if (Calculator::isCritical()) {
120             attackedMessage += CRITICAL_MESSAGE;
121             damage = damage * 2;
122         } else if (Calculator::canDodge(agility)) {
123             attackedMessage += DODGE_MESSAGE;

```

jul 21, 20 15:47

PlayerStats.cpp

Page 3/5

```

122         return {0, true};
123     }
124     int totalDamage = std::max(damage - static_cast<int>(defense), 0);
125     currentLife -= totalDamage;
126     if (currentLife ≤ 0) {
127         currentLife = 0;
128         currentMana = 0;
129     }
130     if (totalDamage > 0) {
131         timeElapsedLife = 0.0;
132     }
133     return {totalDamage, false};
134 }
135
136 int PlayerStats::getMaxLife() const {
137     return maxLife;
138 }
139
140 int PlayerStats::getCurrentLife() const {
141     return currentLife;
142 }
143
144 void PlayerStats::restoreLife(unsigned int amount) {
145     currentLife += static_cast<int>(amount);
146     if (currentLife > maxLife) {
147         currentLife = maxLife;
148     }
149 }
150
151 void PlayerStats::restoreMana(unsigned int amount) {
152     currentMana += amount;
153     if (currentMana > maxMana) {
154         currentMana = maxMana;
155     }
156 }
157
158 bool PlayerStats::isDead() const {
159     return (getCurrentLife() == 0);
160 }
161
162 void PlayerStats::update(double timeStep) {
163     if (isDead()) {
164         return;
165     }
166     timeElapsedLife += timeStep;
167     timeElapsedMana += timeStep;
168     if (timeElapsedLife ≥ TIME_FOR_RECOVERY) {
169         currentLife += Calculator::lifeRecovered(recoveryRate, timeElapsedLife/1
170 000);
171         if (currentLife > maxLife) {
172             currentLife = maxLife;
173         }
174         timeElapsedLife = 0.0;
175     }
176     if (timeElapsedMana ≥ TIME_FOR_RECOVERY) {
177         if (isMeditating) {
178             currentMana += Calculator::manaRecoveredWithMeditation(meditationRat
179 e,
180                                     intelligence,
181                                     timeElapsedMana/1000);
182         } else {
183             currentMana += Calculator::manaRecoveredNoMeditation(recoveryRate,
184                                     timeElapsedMana

```

jul 21, 20 15:47

PlayerStats.cpp

Page 4/5

```

184     if (currentMana ≥ maxMana) {
185         currentMana = maxMana;
186     }
187     timeElapsedMana = 0.0;
188 }
189 }
190
191 void PlayerStats::startMeditating(Minichat& minichat) {
192     if (!isMeditating) {
193         isMeditating = true;
194         minichat.sendMessage(STARTED_MEDITATING_MESSAGE);
195     }
196 }
197
198 void PlayerStats::stopMeditating(Minichat& minichat) {
199     if (isMeditating) {
200         isMeditating = false;
201         minichat.sendMessage(STOPPED_MEDITATING_MESSAGE);
202     }
203 }
204
205 int32_t& PlayerStats::getCurrentMana() {
206     return currentMana;
207 }
208
209 void PlayerStats::restore() {
210     currentMana = maxMana;
211     currentLife = maxLife;
212 }
213
214 void PlayerStats::storeAllRelevantData(std::stringstream& buffer) const {
215     msgpack::type::tuple<int32_t, int32_t, int32_t> xpData(experience, nextLevel
Experience, level);
216     msgpack::pack(buffer, xpData);
217     msgpack::type::tuple<int32_t, int32_t> manaData(currentMana, maxMana);
218     msgpack::pack(buffer, manaData);
219     msgpack::type::tuple<int32_t, int32_t> lifeData(currentLife, maxLife);
220     msgpack::pack(buffer, lifeData);
221     msgpack::type::tuple<int32_t, int32_t, int32_t, int32_t> statsData(strength,
constitution, intelligence, agility);
222     msgpack::pack(buffer, statsData);
223 }
224
225
226 void PlayerStats::storeLifeStatus(std::stringstream& buffer) const {
227     msgpack::type::tuple<bool> isAlive(!isDead());
228     msgpack::pack(buffer, isAlive);
229 }
230
231 void PlayerStats::getData(PlayerData &pData) const {
232     pData.level = level;
233     pData.experience = experience;
234     pData.constitution = constitution;
235     pData.strength = strength;
236     pData.agility = agility;
237     pData.intelligence = intelligence;
238 }
239
240 bool PlayerStats::consumeMana(unsigned int amount) {
241     if (currentMana < static_cast<int>(amount)) {
242         return false;
243     }
244     if (amount ≠ 0) {
245         currentMana -= amount;
246         timeElapsedMana = 0;
247     }
248     return true;

```

jul 21, 20 15:47

PlayerStats.cpp

Page 5/5

```

249 }

```


jul 21, 20 15:47	PlayerProxy.h	Page 1/2
1	<i>//</i>	
2	<i>// Created by marcos on 23/6/20.</i>	
3	<i>//</i>	
4		
5	#ifndef ARGENTUM_PLAYERPROXY_H	
6	#define ARGENTUM_PLAYERPROXY_H	
7		
8	#include <string>	
9	#include <queue>	
10	#include "../Items/Inventory.h"	
11	#include "../Game/Events/Event.h"	
12		
13	class Player;	
14	class Game;	
15	struct Coordinate;	
16		
17	class PlayerProxy {	
18	private:	
19	Game* game{nullptr};	
20	Player* player{nullptr};	
21	std::queue<std::unique_ptr<Event>> storedEvents;	
22		
23	public:	
24	PlayerProxy() = default ;	
25	PlayerProxy(PlayerProxy^ other) noexcept;	
26	PlayerProxy& operator=(PlayerProxy^ other) noexcept;	
27	PlayerProxy(const PlayerProxy& other) = delete;	
28	PlayerProxy& operator=(const PlayerProxy& other) = delete;	
29	explicit PlayerProxy(Game* _game, Player* _player);	
30		
31	<i>/*Encola un ataque*/</i>	
32	void attack(Coordinate target);	
33		
34	<i>/*Encola un use item (equipar para los equipables, consumir para los consumi</i>	
35	<i>bles, etc)*/</i>	
36	void useItem(int32_t itemPosition);	
37		
38	<i>/*Setea al player en modo meditacion (es el unico evento que no encolamos</i>	
39	<i>* porque no valia la pena*/</i>	
40	void meditate();	
41		
42	<i>/*Encola un buy*/</i>	
43	void buyFrom(std::string^ itemName, Coordinate npcPosition);	
44		
45	<i>/*Encola un sell*/</i>	
46	void sellTo(std::string^ itemName, Coordinate npcPosition);	
47		
48	<i>/*Encola un withdraw*/</i>	
49	void withdrawFrom(std::string^ itemName, Coordinate npcPosition);	
50		
51	<i>/*Encola un mensaje a otro player*/</i>	
52	void messageOtherPlayer(std::string^ playerToMessage, std::string^ message)	
53	;	
54		
55	<i>/*Encola un list*/</i>	
56	void listFrom(Coordinate npcPosition);	
57		
58	<i>/*Encola un deposit*/</i>	
59	void depositTo(std::string^ itemName, Coordinate npcPosition);	
60		
61	<i>/*Encola un unequip*/</i>	
62	void unequip(GameType::EquipmentPlace clothing);	
63		
64	<i>/*Encola un drop item*/</i>	
	void dropItem(int32_t itemPosition);	

jul 21, 20 15:47	PlayerProxy.h	Page 2/2
65	<i>/*Encola un pickup de un item*/</i>	
66	void pickUpItem();	
67		
68	<i>/*Encola un resurrect*/</i>	
69	void requestResurrect(Coordinate selectedPosition);	
70		
71	<i>/*Encola un heal*/</i>	
72	void requestHeal(Coordinate selectedPosition);	
73		
74	<i>/*Encola un request para ver el nombre de los items del inventario*/</i>	
75	void getInventoryNames();	
76		
77	<i>/*Guarda en data los datos del player administrado por este PlayerProxy*/</i>	
78	void storeAllRelevantData(std::stringstream& data) const ;	
79		
80	<i>/*Borra el minichat del player*/</i>	
81	void clearMinichat();	
82		
83	<i>/*Mergea la cola del PlayerProxy con la del Game*/</i>	
84	void giveEventsToGame();	
85		
86	<i>/*Encola el comando para empezar a moverse en una direccion*/</i>	
87	void startMoving(GameType::Direction direction);	
88		
89	<i>/*Encola la directiva para dejar de moverse*/</i>	
90	void stopMoving();	
91		
92	<i>/*Retorna los datos actuales del player*/</i>	
93	PlayerData getData() const ;	
94	};	
95		
96		
97	#endif //ARGENTUM_PLAYERPROXY_H	

jul 21, 20 15:47

PlayerProxy.cpp

Page 1/3

```

1  //
2  // Created by marcos on 23/6/20.
3  //
4
5  #include "PlayerProxy.h"
6  #include "../Map/Coordinate.h"
7  #include "../Game/Game.h"
8  #include "../Game/Events/Attack.h"
9  #include "../Game/Events/Buy.h"
10 #include "../Game/Events/Sell.h"
11 #include "../Game/Events/Withdraw.h"
12 #include "../Game/Events/List.h"
13 #include "Player.h"
14 #include "../Game/Events/Deposit.h"
15 #include "../Game/Events/Drop.h"
16 #include "../Game/Events/Unequip.h"
17 #include "../Game/Events/UseItem.h"
18 #include "../Game/Events/PickUpItem.h"
19 #include "../Game/Events/RequestResurrect.h"
20 #include "../Game/Events/Message.h"
21 #include "../Game/Events/RestoreStats.h"
22 #include "../Game/Events/GetInventoryNames.h"
23 #include "../Game/Events/ModifyPlayerMovement.h"
24
25 const unsigned int MAX_EVENTS_STORED = 3;
26
27 PlayerProxy::PlayerProxy(PlayerProxy ^other) noexcept {
28     game = other.game;
29     other.game = nullptr;
30     player = other.player;
31     other.player = nullptr;
32     storedEvents = std::move(other.storedEvents);
33 }
34
35 PlayerProxy &PlayerProxy::operator=(PlayerProxy ^other) noexcept {
36     game = other.game;
37     other.game = nullptr;
38     player = other.player;
39     other.player = nullptr;
40     storedEvents = std::move(other.storedEvents);
41     return *this;
42 }
43
44 PlayerProxy::PlayerProxy(Game *_game, Player *_player) {
45     game = _game;
46     player = _player;
47 }
48
49 void PlayerProxy::attack(Coordinate target) {
50     if (player->getPosition() != target) {
51         if (storedEvents.size() < MAX_EVENTS_STORED) {
52             storedEvents.emplace(new Attack(*player, target));
53         }
54     }
55 }
56
57 void PlayerProxy::useItem(int32_t itemPosition) {
58     if (storedEvents.size() < MAX_EVENTS_STORED) {
59         storedEvents.emplace(new UseItem(*player, itemPosition));
60     }
61 }
62
63 void PlayerProxy::meditate() {
64     player->meditate();
65 }
66

```

jul 21, 20 15:47

PlayerProxy.cpp

Page 2/3

```

67
68 void PlayerProxy::buyFrom(std::string ^itemName, Coordinate npcPosition) {
69     if (storedEvents.size() < MAX_EVENTS_STORED) {
70         storedEvents.emplace(new Buy(*player, std::move(itemName), npcPosition));
71     }
72 }
73
74 void PlayerProxy::sellTo(std::string ^itemName, Coordinate npcPosition) {
75     if (storedEvents.size() < MAX_EVENTS_STORED) {
76         storedEvents.emplace(new Sell(*player, std::move(itemName), npcPosition));
77     }
78 }
79
80 void PlayerProxy::withdrawFrom(std::string ^itemName, Coordinate npcPosition) {
81     if (storedEvents.size() < MAX_EVENTS_STORED) {
82         storedEvents.emplace(new Withdraw(*player, std::move(itemName), npcPosition));
83     }
84 }
85
86 void PlayerProxy::listFrom(Coordinate npcPosition) {
87     if (storedEvents.size() < MAX_EVENTS_STORED) {
88         storedEvents.emplace(new List(*player, npcPosition));
89     }
90 }
91
92 void PlayerProxy::depositTo(std::string ^itemName, Coordinate npcPosition) {
93     if (storedEvents.size() < MAX_EVENTS_STORED) {
94         storedEvents.emplace(new Deposit(*player, std::move(itemName), npcPosition));
95     }
96 }
97
98 void PlayerProxy::unequip(GameType::EquipmentPlace place) {
99     if (storedEvents.size() < MAX_EVENTS_STORED) {
100         storedEvents.emplace(new Unequip(*player, place));
101     }
102 }
103
104 void PlayerProxy::dropItem(int32_t itemPosition) {
105     if (storedEvents.size() < MAX_EVENTS_STORED) {
106         storedEvents.emplace(new Drop(*player, itemPosition));
107     }
108 }
109
110 void PlayerProxy::pickUpItem() {
111     if (storedEvents.size() < MAX_EVENTS_STORED) {
112         storedEvents.emplace(new PickUpItem(*game, *player));
113     }
114 }
115
116 void PlayerProxy::requestResurrect(Coordinate selectedPosition) {
117     if (storedEvents.size() < MAX_EVENTS_STORED) {
118         storedEvents.emplace(new RequestResurrect(*game, *player, selectedPosition));
119     }
120 }
121
122 void PlayerProxy::messageOtherPlayer(std::string ^playerToMessage, std::string ^message) {
123     if (storedEvents.size() < MAX_EVENTS_STORED) {
124         storedEvents.emplace(new Message(*game, *player, std::move(playerToMessage),
125                                         std::move(message)));
126     }
127 }

```

jul 21, 20 15:47

PlayerProxy.cpp

Page 3/3

```

126     }
127 }
128
129 void PlayerProxy::requestHeal(Coordinate selectedPosition) {
130     if (storedEvents.size() < MAX_EVENTS_STORED) {
131         storedEvents.emplace(new RestoreStats(*game, *player, selectedPosition));
132     }
133 }
134
135 void PlayerProxy::getInventoryNames() {
136     if (storedEvents.size() < MAX_EVENTS_STORED) {
137         storedEvents.emplace(new GetInventoryNames(*player));
138     }
139 }
140
141 /*Aca no lo limitamos al tamaño de la cola porque es un evento que no
142 * deberiamos ignorar nunca*/
143 void PlayerProxy::startMoving(GameType::Direction direction) {
144     storedEvents.emplace(new ModifyPlayerMovement(*player, direction));
145 }
146
147 /*Aca no lo limitamos al tamaño de la cola porque es un evento que no
148 * deberiamos ignorar nunca*/
149 void PlayerProxy::stopMoving() {
150     storedEvents.emplace(new ModifyPlayerMovement(*player));
151 }
152
153 void PlayerProxy::giveEventsToGame() {
154     while (!storedEvents.empty()) {
155         game->pushEvent(std::move(storedEvents.front()));
156         storedEvents.pop();
157     }
158 }
159
160 void PlayerProxy::clearMinichat() {
161     player->clearMinichat();
162 }
163
164 void PlayerProxy::storeAllRelevantData(std::stringstream& data) const {
165     player->storeAllRelevantData(data);
166 }
167
168 PlayerData PlayerProxy::getData() const {
169     PlayerData data = player->getData();
170     Game::getPlayerBank(data);
171     return data;
172 }
173
174

```

jul 21, 20 15:47

Player.h

Page 1/3

```

1  //
2  // Created by agustin on 8/6/20.
3  //
4
5  #ifndef ARGENTUM_PLAYER_H
6  #define ARGENTUM_PLAYER_H
7
8
9  #include "../Items/Inventory.h"
10 #include "Entity.h"
11 #include "PlayerStats.h"
12 #include "Minichat.h"
13 #include "../Items/ItemData.h"
14 #include "MovementBackup.h"
15
16 class Game;
17 class EntityTests;
18 class MapTests;
19
20 class Player: public Entity {
21 private:
22     GameType::Race race;
23     GameType::Class pClass;
24     Inventory inventory;
25     PlayerStats stats;
26     int32_t gold;
27     Minichat chat;
28     Game& game;
29     MovementBackup movementBackup{};
30
31     friend EntityTests;
32     friend MapTests;
33
34 public:
35     Player(Game& _game, Coordinate _initialPosition, const PlayerData& data);
36
37     /*Indica si el jugador es target de un monster, un jugador es un target si e
38 sta vivo, si
39 esta muerto no lo es*/
40     bool isMonsterTarget() override;
41
42     /*Ataca el lugar especificado en target*/
43     int32_t attack(Coordinate target) override;
44
45     /*Si hay lugar en el inventario del player entonces se apropia del item y
46 retorna true, sino no se apropia de el y retorna false*/
47     bool storeItem(std::shared_ptr<Item>& item);
48
49     /*Retorna el item que almacene el inventario al pedirle el item con nombre
50 itemName*/
51     std::shared_ptr<Item> removeItem(const std::string& itemName);
52
53     /*Intenta reducir en amount la cantidad de oro que guarda, si esta es menor
54 que amount entonces no la reduce y retorna false, sino la reduce y retorna
55 true*/
56     bool spendGold(int amount);
57
58     /*Incrementa el oro del Player en cantidad amount*/
59     void receiveGold(unsigned int amount);
60
61     /*Usa el item en la posicion indicada, si no hay un item en la posicion no
62 hace nada*/
63     UseReturnData useItem(int itemPosition);
64
65     /*Ataca al player, retorna el danio ocasionado y el xp ganado*/
66     AttackResult attacked(int damage, unsigned int attackerLevel, bool isAPlayer

```

jul 21, 20 15:47	Player.h	Page 2/3
66) override;	
67	/*Restaura hasta amount cantidad de vida, sin pasarse de la cantidad maxima*/	
68	void restoreLife(unsigned int amount);	
69		
70	/*Restaura hasta amount cantidad de mana, sin pasarse de la cantidad maxima*/	
71	void restoreMana(unsigned int amount);	
72		
73	/*Actualiza al player acorde a su estado actual (moviendo, meditando, etc)*/	
74	void update(double timeStep);	
75		
76	/*Hace que el player comience a meditar (el Warrior no puede meditar)*/	
77	void meditate();	
78		
79	/*Desequipa la ropa de equipment place*/	
80	bool unequip(GameType::EquipmentPlace clothing);	
81		
82	/*Compra el item de nombre itemName del npc en la posicion npcPosition, si n	
83	o hay	
84	uno no hace nada*/	
85	void buyFrom(const std::string& itemName, Coordinate npcPosition);	
86	/*Vende el item de nombre itemName al npc en la posicion npcPosition, si no	
87	hay	
88	uno no hace nada*/	
89	void sellTo(const std::string& itemName, Coordinate npcPosition);	
90		
91	/*Pide recuperar un item de una posicion, en el caso que haya un banker	
92	* se retornara el item pedido (si existiera), si no hay un banker no pasa	
93	* nada*/	
94	void withdrawFrom(const std::string& itemName, Coordinate npcPosition);	
95		
96	/*Pide que le listen los items a un NPC ubicado en npcPosition, si no hay	
97	* un npc valido no pasa nada*/	
98	void listFrom(Coordinate npcPosition);	
99		
100	/*Intenta depositar un item en el banker ubicado en npcPosition, si no hay	
101	* un npc valido no pasa nada*/	
102	void depositTo(const std::string& itemName, Coordinate npcPosition);	
103		
104	/*Desequipa el arma*/	
105	bool unequip();	
106		
107	/*Retorna una instancia de un ItemData que guarda el tipo e id de item y la	
108	posicion en la que el item fue dejado, si no se encuentra un item en la posi	
109	cion	
110	recibida entonces se almacena -1 en el id del item*/	
111	const Item* dropItem(unsigned int itemPosition);	
112		
113	/*Carga los datos generales del player de acuerdo al protocolo, se utiliza	
114	* para la info que se le envia a un cliente recién conectado*/	
115	void operator>>(std::stringstream& buffer) const override;	
116		
117	/*Almacena los datos personales del player acorde al protocolo, estos	
118	* datos solo son enviados al cliente que controla a ese player	
119	* (seria la UI externa al mapa)*/	
120	void storeAllRelevantData(std::stringstream& buffer) const;	
121		
122	/*Limpia el minichat del player, se llama despues de haber enviado el update	
123	* al cliente que controla a ese player. Es para no mandar los mensajes repe	
124	tidos*/	
	void clearMinichat();	
	/*Agrega un mensaje al minichat del player*/	

jul 21, 20 15:47	Player.h	Page 3/3
125	void addMessage(const std::string& message);	
126		
127	/*Le confirma a entity el request de movimiento para comenzar la interpolaci	
128	on*/	
129	void move(Coordinate newPosition) override;	
130		
131	/*Setea el mana y la vida del player al maximo*/	
132	void restoreStats(bool isBeingRevived);	
133		
134	/*Retorna true si el player esta muerto, false en caso contrario*/	
135	bool isDead();	
136		
137	/*Resetea la interpolacion del player*/	
138	void resetMovement();	
139		
140	/*Retorna true si el player tiene dicho item, false en caso contrario*/	
141	bool hasItem(const std::string& itemName);	
142		
143	/*Agrega al minichat del player en nombre de los items que tiene en el	
144	* inventario para que el cliente sepa como se llaman*/	
145	void getInventoryNames();	
146		
147	/*Setea la direccion de movimiento del player en direction*/	
148	void startMovement(GameType::Direction direction);	
149		
150	/*Deja de mover al player*/	
151	void stopMovement();	
152		
153	int32_t getLevel() const override;	
154		
155	bool hasFullInventory() const;	
156		
157	/*Retorna su data actual (las cosas guardadas, stats, etc)*/	
158	PlayerData getData() const;	
159		
160	private:	
161	void _dropItems();	
162	void _storeAttackedResultMessage(std::string& resultMessage, std::pair<int,	
163	bool> attackResult,	
164	unsigned int experience);	
165	AttackResult _receiveDamage(int damage, unsigned int attackerLevel, bool isA	
166	Player);	
167	};	
	#endif //ARGENTUM_PLAYER_H	

jul 21, 20 15:47

Player.cpp

Page 1/6

```

1  //
2  // Created by agustin on 8/6/20.
3  //
4
5  #include "Player.h"
6  #include "../Config/Calculator.h"
7  #include "AttackResult.h"
8  #include "../Game/Game.h"
9  #include "../Items/Miscellaneous/Gold.h"
10 #include "../Game/Events/Drop.h"
11 #include "../Game/Events/NotifyDeath.h"
12 #include "../Game/Events/Move.h"
13 #include "../Game/Events/PlayerLeveledUp.h"
14 #include <msgpack.hpp>
15
16 #define ATTACKER_IS_NEWBIE_MESSAGE "I won't lose my time on a low level newbie like you!\n"
17 #define PLAYER_IS_A_NEWBIE_MESSAGE "Surely you have better things to do than attack a low level newbie like me...\n"
18 #define PLAYER_IS_DEAD_MESSAGE "You can't kill a ghost, you know?\n"
19 #define DODGED_ATTACK_MESSAGE "You dodged an attack\n"
20 #define GHOSTS_CANT_RESTORE_STATS_MESSAGE "Ghosts can't restore stats\n"
21 #define WARRIOR_CANT_MEDITATE_MESSAGE "You are a Warrior and Warriors cannot meditate!\n"
22
23 using namespace GameType;
24
25 MSGPACK_ADD_ENUM(GameType::Race)
26 MSGPACK_ADD_ENUM(GameType::EventID)
27
28 ////////////////////////////////////PUBLIC////////////////////////////////////
29
30 Player::Player(Game& _game, Coordinate _initialPosition, const PlayerData& data)
31 :
32     Entity(GameType::Entity::PLAYER, _initialPosition, data.nickname,
33           true),
34     inventory(data),
35     stats(data),
36     game(_game) {
37
38     speed = Configuration::getInstance().configPlayerSpeed();
39     pClass = data.pClass;
40     race = data.pRace;
41     gold = data.gold;
42     movementBackup = {false, GameType::DIRECTION_STILL};
43 }
44
45 int32_t Player::attack(Coordinate target) {
46     int32_t returnValue = -1;
47     if (!stats.isDead()) {
48         stats.stopMeditating(chat);
49         int weaponDamage;
50         weaponDamage = inventory.getWeaponDamage(currentPosition, target, stats)
51 ;
52         int totalDamage = stats.getTotalDamage(weaponDamage);
53         if (totalDamage != 0) {
54             std::pair<AttackResult, bool> result = game.attackPosition(totalDamage, stats.getLevel(),
55                               true, target);
56             if (stats.increaseExperience(result.first.experience)) {
57                 game.pushEvent(std::unique_ptr<Event>(new PlayerLeveledUp(getNickname(), stats.getLevel())));
58             }
59             chat.addMessage(std::move(result.first.resultMessage));
60             if (result.second) {
61                 returnValue = inventory.getWeaponId();
62             }
63         }
64     }
65 }

```

jul 21, 20 15:47

Player.cpp

Page 2/6

```

61 }
62 return returnValue;
63 }
64
65
66 AttackResult Player::attacked(int damage, unsigned int attackerLevel, bool isAPlayer) {
67     stats.stopMeditating(chat);
68     unsigned int newbieLevel = Configuration::getInstance().configNewbieLevel();
69     if (!stats.isDead()) {
70         if (isAPlayer ^ damage > 0) {
71             if (stats.getLevel() ≤ newbieLevel) {
72                 return {0, 0, PLAYER_IS_A_NEWBIE_MESSAGE};
73             } else if (attackerLevel ≤ newbieLevel) {
74                 return {0, 0, ATTACKER_IS_NEWBIE_MESSAGE};
75             }
76         }
77         return _receiveDamage(damage, attackerLevel, isAPlayer);
78     } else {
79         return {0, 0, PLAYER_IS_DEAD_MESSAGE};
80     }
81 }
82
83 bool Player::isMonsterTarget() {
84     return !stats.isDead();
85 }
86
87 bool Player::spendGold(int amount) {
88     stats.stopMeditating(chat);
89     if ((!stats.isDead()) ^ (amount ≤ gold)) {
90         gold -= amount;
91         return true;
92     }
93     return false;
94 }
95
96 void Player::receiveGold(unsigned int amount) {
97     unsigned int maxGold = Calculator::calculateMaxSafeGold(stats.getLevel());
98     maxGold += maxGold / 2;
99     if (!stats.isDead() ^ (gold + amount) ≤ maxGold) {
100         gold += amount;
101     }
102 }
103
104 bool Player::storeItem(std::shared_ptr<Item> &item) {
105     if ((!stats.isDead()) ^ (item)) {
106         stats.stopMeditating(chat);
107         if (item->isGold()) {
108             std::shared_ptr<Gold> aux = std::dynamic_pointer_cast<Gold>(item);
109             receiveGold(aux->getAmount());
110             return true;
111         } else {
112             return inventory.addItem(item);
113         }
114     }
115     return false;
116 }
117
118 std::shared_ptr<Item> Player::removeItem(const std::string &itemName) {
119     if (!stats.isDead()) {
120         stats.stopMeditating(chat);
121         return inventory.removeItem(itemName);
122     }
123     return nullptr;
124 }
125

```

jul 21, 20 15:47

Player.cpp

Page 3/6

```

126 UseReturnData Player::useItem(int itemPosition) {
127     if (!stats.isDead()) {
128         stats.stopMeditating(chat);
129         return inventory.useItem(*this, itemPosition);
130     }
131     return {GameType::EQUIPMENT_PLACE_NONE, -1};
132 }
133
134 void Player::restoreLife(unsigned int amount) {
135     if (!stats.isDead()) {
136         stats.restoreLife(amount);
137     } else {
138         chat.sendMessage(GHOSTS_CANT_RESTORE_STATS_MESSAGE);
139     }
140 }
141
142 void Player::restoreMana(unsigned int amount) {
143     if (!stats.isDead()) {
144         stats.restoreMana(amount);
145     } else {
146         chat.sendMessage(GHOSTS_CANT_RESTORE_STATS_MESSAGE);
147     }
148 }
149
150 void Player::meditate() {
151     if (!stats.isDead()) {
152         if (pClass != GameType::WARRIOR) {
153             stats.startMeditating(chat);
154         } else {
155             addMessage(WARRIOR_CANT_MEDITATE_MESSAGE);
156         }
157     }
158 }
159
160 void Player::update(double timeStep) {
161     Entity::update(timeStep, game); /*actualiza movimiento*/
162     if (!movement.isMoving ^ movementBackup.isFollowingRoad) {
163         game.pushEvent(std::unique_ptr<Event>(new Move(game, *this,
164             movementBackup.direction)));
165     }
166     stats.update(timeStep); /*actualiza la vida y manda en base al tiempo/medita
167     cion*/
168 }
169
170 bool Player::unequip(EquipmentPlace clothing) {
171     if (!stats.isDead()) {
172         stats.stopMeditating(chat);
173         return inventory.unequip(clothing);
174     }
175     return false;
176 }
177
178 bool Player::unequip() {
179     if (!stats.isDead()) {
180         stats.stopMeditating(chat);
181         return inventory.unequip();
182     }
183     return false;
184 }
185
186 const Item* Player::dropItem(unsigned int itemPosition) {
187     std::shared_ptr<Item> aux = inventory.removeItem(itemPosition);
188     const Item* returnData = aux.get();
189     if (aux) {
190         game.dropItems(std::move(aux), currentPosition);
191     }

```

jul 21, 20 15:47

Player.cpp

Page 4/6

```

191     return returnData;
192 }
193
194 void Player::buyFrom(const std::string &itemName, Coordinate npcPosition) {
195     game.buy(*this, itemName, npcPosition);
196 }
197
198 void Player::sellTo(const std::string &itemName, Coordinate npcPosition) {
199     game.sell(*this, itemName, npcPosition);
200 }
201
202 void Player::withdrawFrom(const std::string &itemName, Coordinate npcPosition) {
203     game.withdraw(*this, itemName, npcPosition);
204 }
205
206 void Player::listFrom(Coordinate npcPosition) {
207     game.list(*this, npcPosition);
208 }
209
210 void Player::depositTo(const std::string &itemName, Coordinate npcPosition) {
211     game.deposit(*this, itemName, npcPosition);
212 }
213
214 void Player::operator>>(std::stringstream &buffer) const {
215     Entity::operator>>(buffer);
216     msgpack::type::tuple<Race> data(race);
217     msgpack::pack(buffer, data);
218     stats.storeLifeStatus(buffer);
219     inventory.storeEquippedItems(buffer);
220 }
221
222 void Player::storeAllRelevantData(std::stringstream& buffer) const {
223     msgpack::type::tuple<int32_t, int32_t> data(gold, Calculator::calculateMaxSa
224     feGold(stats.getLevel()));
225     msgpack::pack(buffer, data);
226     inventory.storeAllData(buffer);
227     stats.storeAllRelevantData(buffer);
228     msgpack::type::tuple<int32_t, int32_t> position(currentPosition.iPosition,
229     currentPosition.jPosition);
230     msgpack::pack(buffer, position);
231     msgpack::type::tuple<std::string> minichat(chat.getMessages());
232     msgpack::pack(buffer, minichat);
233     msgpack::type::tuple<std::string> nick(Entity::getNickname());
234     msgpack::pack(buffer, nick);
235 }
236
237 void Player::clearMinichat() {
238     chat.clear();
239 }
240
241 void Player::addMessage(const std::string &message) {
242     chat.addMessage(message);
243 }
244
245 void Player::move(Coordinate newPosition) {
246     stats.stopMeditating(chat);
247     Entity::move(newPosition);
248 }
249
250 void Player::restoreStats(bool isBeingRevived) {
251     //if ((esta vivo y no pide que lo resuciten) || (pide que lo resuciten y est
252     a muerto))
253     //Esta funcion se usa para curar y para cuando se resucite, se chequea para
254     que no se cure
255     //cuando llame a resucitar si es que esta vivo y para que no resucite si est
256     a vivo

```

jul 21, 20 15:47

Player.cpp

Page 5/6

```

253     if ((¬stats.isDead() ^ ¬isBeingRevived) ∨ (isBeingRevived ^ stats.isDead()))
254     ) {
255         stats.restore();
256     }
257     if (stats.isDead() ^ ¬isBeingRevived) {
258         chat.addMessage(GHOSTS_CANT_RESTORE_STATS_MESSAGE);
259     }
260 }
261 bool Player::isDead() {
262     return stats.isDead();
263 }
264 void Player::resetMovement() {
265     movement.movedDistance = 0;
266     movement.direction = DIRECTION_STILL;
267     movement.isMoving = false;
268 }
269 bool Player::hasItem(const std::string& itemName) {
270     return inventory.hasItem(itemName);
271 }
272 void Player::getInventoryNames() {
273     inventory.getInventoryNames(chat);
274 }
275 void Player::startMovement(GameType::Direction direction) {
276     movementBackup.isFollowingRoad = true;
277     movementBackup.direction = direction;
278 }
279 void Player::stopMovement() {
280     movementBackup.isFollowingRoad = false;
281 }
282 PlayerData Player::getData() const {
283     PlayerData pData;
284     pData.nickname = getNickname();
285     pData.pRace = race;
286     pData.pClass = pClass;
287     pData.gold = gold;
288     stats.getData(pData);
289     inventory.getData(pData);
290     return pData;
291 }
292 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
293 void Player::_dropItems() {
294     std::list<std::shared_ptr<Item>> items = inventory.dropAllItems();
295     int goldDropped = static_cast<int>(gold -
296                                     Calculator::calculateMaxSafeGold(stats.ge
297                                     tLevel()));
298     goldDropped = std::max(goldDropped, 0);
299     gold -= goldDropped;
300     if (goldDropped > 0) {
301         items.emplace_back(std::make_shared<Gold>(goldDropped));
302     }
303     if (¬items.empty()) {
304         //game.dropItems(std::move(items), currentPosition);
305         game.pushEvent(std::unique_ptr<Event>(new Drop(game, std::move(items), c
306         urrentPosition)));
307     }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }

```

jul 21, 20 15:47

Player.cpp

Page 6/6

```

316 void Player::_storeAttackedResultMessage(std::string& resultMessage, std::pair<i
317 nt, bool> attackResult,
318         unsigned int experience) {
319     std::string damageString = std::to_string(std::abs(attackResult.first));
320     if (attackResult.second) {
321         resultMessage += getNickname() + " dodged your attack\n";
322     } else if (attackResult.first ≥ 0) {
323         resultMessage += "You damaged " + getNickname() + " by " + damageString;
324         resultMessage += " (Remaining Life: " + std::to_string(stats.getCurrentLife()
325         ) +
326         ", XP Gained: " + std::to_string(experience) + ")\n";
327     } else {
328         resultMessage += "You healed " + getNickname() + " by " + damageString;
329         resultMessage += " (Remaining Life: " + std::to_string(stats.getCurrentLife()
330         ) + ")\n";
331     }
332     //Se agrega el mensaje del ataque al minichat del atacado
333     if (attackResult.second) {
334         chat.addMessage(DODGED_ATTACK_MESSAGE);
335     } else if (attackResult.first ≥ 0) {
336         chat.addMessage("You lost " + damageString + " health points\n");
337     } else if (attackResult.first < 0) {
338         chat.addMessage("You healed " + damageString + " health points\n");
339     }
340 }
341 AttackResult Player::_receiveDamage(int damage, unsigned int attackerLevel, bool
342 isAPlayer) {
343     std::string attackedMessage;
344     unsigned int defense = inventory.getDefense();
345     std::pair<int, bool> result = stats.modifyLife(damage, attackerLevel, defens
346     e, isAPlayer, attackedMessage);
347     unsigned int experience = Calculator::calculateAttackXP(result.first,
348         attackerLevel, stats
349         .getLevel());
350     if (stats.isDead() ^ result.first > 0) {
351         _dropItems();
352         experience += Calculator::calculateKillXP(attackerLevel,
353             stats.getLevel(), stats.getMax
354             Life());
355         game.pushEvent(std::unique_ptr<Event>(new NotifyDeath(*this)));
356     }
357     _storeAttackedResultMessage(attackedMessage, result, experience);
358     return {result.first, experience, std::move(attackedMessage)};
359 }
360 int32_t Player::getLevel() const {
361     return stats.getLevel();
362 }
363 bool Player::hasFullInventory() const {
364     return inventory.isFull();
365 }

```

jul 21, 20 15:47

MovementBackup.h

Page 1/1

```

1 //
2 // Created by agustin on 14/7/20.
3 //
4
5 #ifndef ARGENTUM_MOVEMENTBACKUP_H
6 #define ARGENTUM_MOVEMENTBACKUP_H
7
8 #include "../libs/GameEnums.h"
9
10 struct MovementBackup {
11     bool isFollowingRoad;
12     GameType::Direction direction;
13 };
14
15 #endif //ARGENTUM_MOVEMENTBACKUP_H

```

jul 21, 20 15:47

MonsterStats.h

Page 1/1

```

1 //
2 // Created by marcos on 20/6/20.
3 //
4
5 #ifndef ARGENTUM_MONSTERSTATS_H
6 #define ARGENTUM_MONSTERSTATS_H
7
8 #include "../libs/GameEnums.h"
9 #include <string>
10
11 class EntityTests;
12 struct AttackResult;
13
14 class MonsterStats {
15 private:
16     unsigned int constitution;
17     unsigned int agility;
18     unsigned int strength;
19     int damage;
20     unsigned int speed;
21
22     unsigned int level;
23     int currentLife;
24     int maxLife;
25     unsigned int rangeOfVision;
26
27     friend EntityTests;
28
29 public:
30     explicit MonsterStats(GameType::Entity type);
31
32     /*Retorna el rango de vision del monster*/
33     unsigned int getRangeOfVision() const;
34
35     /*Retorna el danio del monster*/
36     int getDamage() const;
37
38     /*Retorna el nivel del monster*/
39     unsigned int getLevel() const;
40
41     /*Recibe el danio y modifica la vida del monster acorde a este, retorna luego
42     o
43     * un pair con el danio neto recibido y un bool en true en caso de que haya
44     * logrado esquivar el ataque (esto es asi ya que queda extendible por si el
45     * monstruo pudiera tener defensa y el danio fuera 0 sin haber esquivado)
46     * Almacena en attackedMessage prefijos para el mensaje a mostrar segun el
47     * resultado del ataque*/
48     std::pair<int, bool> modifyLife(int _damage, std::string& attackedMessage);
49
50     /*Retorna la vida actual del monstruo*/
51     int getCurrentLife() const;
52
53     /*Retorna la vida maxima del monstruo*/
54     int getMaxLife() const;
55
56     /*Retorna la agilidad del monstruo*/
57     unsigned int getAgility() const;
58 };
59
60 #endif //ARGENTUM_MONSTERSTATS_H

```


jul 21, 20 15:47

MonsterStats.cpp

Page 1/1

```

1 //
2 // Created by marcos on 20/6/20.
3 //
4
5 #include "MonsterStats.h"
6 #include "../Config/Configuration.h"
7 #include "../Config/Calculator.h"
8
9 #define CRITICAL_ATTACK_MESSAGE "Critical attack. "
10 #define DODGED_ATTACK_MESSAGE "Fear runs through your spine. "
11
12 MonsterStats::MonsterStats(GameType::Entity type) {
13     Configuration& config = Configuration::getInstance();
14     Config::MonsterStats stats = config.configMonsterStats(type);
15     level = Calculator::getRandomInt(static_cast<int>(stats.minLevel),
16                                     static_cast<int>(stats.maxLevel));
17     constitution = stats.constitution + level;
18     strength = stats.strength + level;
19     agility = stats.agility + level;
20     damage = static_cast<int>(stats.damage) + 10 * strength;
21     rangeOfVision = stats.rangeOfVision;
22     maxLife = stats.life + constitution * 10;
23     currentLife = maxLife;
24     speed = stats.speed;
25 }
26
27 unsigned int MonsterStats::getRangeOfVision() const {
28     return rangeOfVision;
29 }
30
31 int MonsterStats::getDamage() const {
32     return damage;
33 }
34
35 unsigned int MonsterStats::getLevel() const {
36     return level;
37 }
38
39 std::pair<int, bool> MonsterStats::modifyLife(int _damage, std::string& attacked
40 Message) {
41     if (Calculator::isCritical()) {
42         _damage *= 2;
43         attackedMessage += CRITICAL_ATTACK_MESSAGE;
44     } else if (Calculator::canDodge(getAgility())) {
45         attackedMessage += DODGED_ATTACK_MESSAGE;
46         return {0, true};
47     }
48     currentLife -= _damage;
49     if (currentLife < 0) {
50         currentLife = 0;
51     }
52     return {_damage, false};
53 }
54
55 int MonsterStats::getCurrentLife() const {
56     return currentLife;
57 }
58
59 int MonsterStats::getMaxLife() const {
60     return maxLife;
61 }
62
63 unsigned int MonsterStats::getAgility() const {
64     return agility;
65 }

```

jul 21, 20 15:47

Monster.h

Page 1/2

```

1 //
2 // Created by agustin on 7/6/20.
3 //
4
5 #ifndef ARGENTUM_MONSTER_H
6 #define ARGENTUM_MONSTER_H
7
8
9 #include "Entity.h"
10 #include <memory>
11 #include "MonsterStats.h"
12 #include "../libs/GameEnums.h"
13
14 class Game;
15 class Map;
16 class Item;
17 class EntityTests;
18
19 class Monster: public Entity {
20 private:
21     const unsigned int timeBetweenActions;
22     double elapsedTime;
23     int inactiveCycles;
24     MonsterStats stats;
25     GameType::Weapon monsterWeapon;
26
27     /*Guarda parte del camino para no llamar al pathfinding cada vez que se
28     quiera mover*/
29     std::list<Coordinate> pathCache;
30
31     /*Guarda el mapa para ver sus alrededores, el juego lo guarda para poder
32     modificarlo*/
33     const Map& map;
34     Game& game;
35
36     friend EntityTests;
37
38 private:
39     static unsigned int _getDistance(Coordinate a, Coordinate b);
40
41     void _storeNearestPlayerPathCache();
42
43     bool _tryToAttack();
44
45     void _move();
46
47 public:
48     Monster(Game& _game, Coordinate initialPositionunsigned,
49             GameType::Entity _type, GameType::Weapon _weapon);
50
51
52     /*Daña a el monstruo, retorna la cantidad de danio recibido
53     Recibe level por un tema de herencia y logica del juego, esto igualmente
54     hace que el ataque sea extendible, podria aplicarse la logica del fair
55     play tambien a los monstruos. El isAPlayer no lo usa pero hacia falta
56     para el polimorfismo de Player*/
57     AttackResult attacked(int damage, unsigned int attackerLevel, bool isAPlayer
58 ) override;
59
60     /*Actualiza al monstruo acorde a su estado (por ejemplo si esta moviendose
61     * actualiza la interpolacion, o si tiene a un playera la vista lo
62     * ataca*/
63     void update(double timeStep);
64
65     /*Indica si el monstruo esta muerto, retorna true si lo esta, sino retorna f
66     else*/

```

jul 21, 20 15:47

Monster.h

Page 2/2

```

65     bool isDead() const;
66
67     /*Ataca una posicion (lo hace cuando tiene un player a la vista*/
68     int32_t attack(Coordinate attackedPosition) override;
69
70     int32_t getLevel() const override;
71 };
72
73
74 #endif //ARGENTUM_MONSTER_H

```

jul 21, 20 15:47

Monster.cpp

Page 1/3

```

1  //
2  // Created by agustin on 7/6/20.
3  //
4
5  #include "Monster.h"
6  #include "../Items/ItemsFactory.h"
7  #include "../Game/Game.h"
8  #include "AttackResult.h"
9  #include "../Config/Configuration.h"
10 #include "../Game/Events/Attack.h"
11 #include "../Game/Events/Move.h"
12 #include "../Game/Events/Drop.h"
13 #include "../Config/Calculator.h"
14
15 #define MAX_NUMBER_OF_CACHED_NODES 4
16
17 //////////////////////////////////PRIVATE////////////////////////////////////
18
19 //Retorna la distancia (siempre positiva) entre las dos coordenadas
20 unsigned int Monster::_getDistance(Coordinate a, Coordinate b) {
21     return std::abs(a.iPosition - b.iPosition) + std::abs(a.jPosition - b.jPosition);
22 }
23
24 /*Guarda parte del camino al jugador al cual tiene que moverse la menor cantidad
25 de veces para alcanzarlo*/
26 void Monster::_storeNearestPlayerPathCache() {
27     unsigned int nearestTargetIndex = 0;
28     std::vector<Coordinate> positions;
29     map.getMoveTargets(currentPosition, stats.getRangeOfVision(), positions);
30     if (!positions.empty()) {
31         std::vector<std::list<Coordinate>> allPaths/(positions.size()*)/;
32         std::list<Coordinate> aux;
33         for (auto & position : positions) {
34             if (map.getPath(currentPosition, position, aux)) {
35                 allPaths.push_back(std::move(aux));
36                 if (allPaths[allPaths.size() - 1].size() < allPaths[nearestTargetIndex].size()) {
37                     nearestTargetIndex = allPaths.size() - 1;
38                 }
39                 aux.clear();
40             }
41         }
42         if (!allPaths.empty()) {
43             pathCache = std::move(allPaths[nearestTargetIndex]);
44             if (pathCache.size() > MAX_NUMBER_OF_CACHED_NODES) {
45                 pathCache.resize(MAX_NUMBER_OF_CACHED_NODES);
46             }
47         }
48     }
49 }
50
51 /*Intenta atacar en sus alrededores, si no encuentra un jugador a quien atacar
52 no hace nada y retorna false, sino vacia pathCache, ataca y retorna true*/
53 bool Monster::_tryToAttack() {
54     std::vector<Coordinate> targets;
55     map.getAttackTargets(currentPosition, stats.getRangeOfVision(), targets);
56     for (auto & target : targets) {
57         if (_getDistance(currentPosition, target) == 1) {
58             std::unique_ptr<Attack> attackFunction(new Attack(*this, target));
59             game.pushEvent(std::move(attackFunction));
60             pathCache.clear();
61             inactiveCycles = 0;
62             return true;
63         }
64     }

```

jul 21, 20 15:47

Monster.cpp

Page 2/3

```

65     return false;
66 }
67
68 /*Pide al game que lo mueva a la siguiente posicion en pathCache, si pathCache
69 esta vacio entonces busca el jugador mas cercano en su rango de vision y le
70 pide al mapa un camino a este
71 Si la proxima posicion a la que se va a mover esta ocupada entonces vuelve a
72 calcular el camino al jugador mas cercano (esto puede pasar si un monstruo se
73 pone en su camino)*/
74 void Monster::_move() {
75     if (!map.isPlaceAvailable(pathCache.front())) {
76         pathCache.clear();
77     }
78     if (pathCache.empty()) {
79         _storeNearestPlayerPathCache();
80     }
81     if (!pathCache.empty()) {
82         game.pushEvent(std::unique_ptr<Move>(new Move(game, *this,
83             _getMoveDirection(pathCache.front()))));
84         pathCache.pop_front();
85         inactiveCycles = 0;
86     } else if (inactiveCycles ≥ 10) {
87         Coordinate newPosition = map.getMonsterRandomPosition(currentPosition);
88         Coordinate noPositions = {-1, -1};
89         if (newPosition ≠ noPositions) {
90             game.pushEvent(std::unique_ptr<Move>(new Move(game, *this,
91                 _getMoveDirection(newPosition))));
92         }
93         inactiveCycles = 0;
94     }
95 }
96
97 ///////////////////////////////////////////////////PUBLIC////////////////////////////////////
98
99
100 Monster::Monster(Game &game, Coordinate initialPosition,
101     GameType::Entity _type, GameType::Weapon _weapon):
102     Entity(_type, initialPosition, "Monster"),
103     timeBetweenActions(Configuration::getInstance().configMonsterSt
104 ats(_type).reactionSpeed * 200),
105     stats(_type), map(_game.getMap()), game(_game) {
106     monsterWeapon = _weapon;
107     elapsedTime = 0;
108     inactiveCycles = 0;
109     type = _type;
110     speed = Configuration::getInstance().configMonsterStats(_type).speed;
111 }
112
113 AttackResult Monster::attacked(int _damage, unsigned int attackerLevel, bool isA
114 Player) {
115     AttackResult result{0, 0, ""};
116     if (_damage ≤ 0) return result;
117     if (!isDead()) {
118         std::pair<int, bool> realAttackResult = stats.modifyLife(_damage, result
119 .resultMessage);
120         if (realAttackResult.second) {
121             result.resultMessage += "The monster dodged your attack\n";
122         } else {
123             unsigned int experience = Calculator::calculateKillXP(attackerLevel,
124                 stats.getLevel(), stats.getMaxLife());
125             result.damage = realAttackResult.first;
126             result.experience = experience;
127             result.resultMessage += "You damaged the Monster by " +
128                 std::to_string(result.damage) + " (Remaining Lif
129 e: " +
130                 std::to_string(stats.getCurrentLife()) +

```

jul 21, 20 15:47

Monster.cpp

Page 3/3

```

127         nce) + "\n";
128     }
129     if (isDead()) {
130         std::shared_ptr<Item> drop;
131         ItemsFactory::getInstance().storeRandomDrop(drop, stats.getMaxLife()
132 );
133         if (drop) {
134             game.pushEvent(std::unique_ptr<Event>(new Drop(game,
135                 std::move(drop), currentPosition)));
136         }
137     }
138     return result;
139 }
140
141 void Monster::update(double timeStep) {
142     Entity::update(timeStep, game);
143     elapsedTime += timeStep;
144     if (elapsedTime ≥ timeBetweenActions) {
145         elapsedTime = 0;
146         ++inactiveCycles;
147         if (!tryToAttack() ^ !isMoving()) {
148             _move();
149         }
150     }
151 }
152
153 bool Monster::isDead() const {
154     return (stats.getCurrentLife() == 0);
155 }
156
157 int32_t Monster::attack(Coordinate attackedPosition) {
158     game.attackPosition(stats.getDamage(), stats.getLevel(), false, attackedPosi
159 tion);
160     return monsterWeapon;
161 }
162
163 int32_t Monster::getLevel() const {
164     return stats.getLevel();
165 }

```

jul 21, 20 15:47

Minichat.h

Page 1/1

```

1  //
2  // Created by marcos on 7/3/20.
3  //
4
5  #ifndef ARGENTUM_MINICHAT_H
6  #define ARGENTUM_MINICHAT_H
7
8  #include <string>
9
10 class Minichat {
11 private:
12     std::string message;
13
14 public:
15     /*Agrega un mensaje al minichat del player*/
16     void addMessage(std::string^ msg);
17     void addMessage(const std::string& msg);
18
19     /*Retorna el string del minichat (el minichat en si digamos)*/
20     std::string getMessages() const;
21
22     /*Limpia el minichat (elimina el string)*/
23     void clear();
24 };
25
26
27 #endif //ARGENTUM_MINICHAT_H

```

jul 21, 20 15:47

Minichat.cpp

Page 1/1

```

1  //
2  // Created by marcos on 7/3/20.
3  //
4
5  #include "Minichat.h"
6
7  void Minichat::addMessage(const std::string &msg) {
8      message += msg;
9  }
10
11 void Minichat::addMessage(std::string^ msg) {
12     message += msg;
13 }
14
15 std::string Minichat::getMessages() const {
16     return message;
17 }
18
19 void Minichat::clear() {
20     message.clear();
21 }

```

jul 21, 20 15:47	Entity.h	Page 1/3
<pre> 1 // 2 // Created by agustin on 6/6/20. 3 // 4 5 #ifndef ARGENTUM_ENTITY_H 6 #define ARGENTUM_ENTITY_H 7 8 9 #include "../Map/Coordinate.h" 10 #include <list> 11 #include <chrono> 12 #include "../libs/GameEnums.h" 13 14 struct ProductData; 15 struct AttackResult; 16 class Game; 17 18 struct Movement { 19 bool isMoving{false}; 20 unsigned int movedDistance{0}; 21 22 /*Esta direccion solo tiene sentido si se setea que se esta moviendo el enti 23 ty*/ 24 GameType::Direction direction{GameType::DIRECTION_STILL}; 25 }; 26 27 class Player; 28 29 class Entity { 30 private: 31 static unsigned int availableId; 32 std::string nickname; 33 34 protected: 35 Coordinate currentPosition{}; 36 GameType::Entity type; 37 Movement movement{}; 38 unsigned int speed{}; 39 40 private: 41 Coordinate _calculatePreviousPosition() const; 42 43 protected: 44 GameType::Direction _getMoveDirection(Coordinate destination); 45 46 public: 47 Entity(GameType::Entity _type, Coordinate initialPosition, const std::string 48 & _nicknamePrefix, 49 bool isPrefixUnique = false); 50 51 /*Implementa el comportamiento realizado al atacar, 52 debe ser modificado en las clases hijas de ser necesario*/ 53 virtual int32_t attack(Coordinate target); 54 55 /*Implementa el comportamiento realizado al ser atacado, por default 56 retorna struct nulo, debe ser modificado en las clases hijas de ser necesari 57 o*/ 58 virtual AttackResult attacked(int damage, unsigned int level, bool isAPlayer 59); 60 61 /*Implementa el comportamiento realizado al pedirle una lista de los items 62 que tiene en venta, por default no hace nada, debe ser reimplementada 63 si la clase hija tiene objetos en venta para listar*/ 64 virtual void list(Player &player); 65 66 /*Implementa el comportamiento realizado al pedirle uno de los items </pre>	<pre> 63 que tiene guardados, por default no hace nada, debe ser reimplementada 64 si la clase hija puede almacenar y devolver items*/ 65 virtual void withdraw(Player& player, const std::string& itemName); 66 67 /*Implementa el comportamiento realizado al pedirle que guarde el item que s 68 e 69 encuentra en la posicion pasada, por default no hace nada, debe ser reimplem 70 entada 71 si la clase hija puede almacenar y devolver items*/ 72 virtual void deposit(Player& player, const std::string& itemName); 73 74 /*Implementa el comportamiento realizado al intentar comprar un item 75 con el nombre pasado, por default no hace nada, debe ser reimplementada 76 si la clase hija puede comprar y vender items*/ 77 virtual void buy(Player& player, const std::string& itemName); 78 79 /*Implementa el comportamiento realizado al intentar vender un item que se 80 encuentra en la posicion pasada, por default no hace nada, debe ser reimplem 81 entada 82 si la clase hija puede comprar y vender items*/ 83 virtual void sell(Player& player, const std::string& itemName); 84 85 /*Indica si el Entity va a ser atacado por un monstruo, por default retorna 86 false, las clases que hereden deben reimplementarla si son consideradas 87 targets*/ 88 virtual bool isMonsterTarget(); 89 90 /*Le asigna al jugador la posicion recibida*/ 91 void setPosition(Coordinate coordinate); 92 93 /*Retorna la posicion en la que quiere estar el jugadoras*/ 94 Coordinate getPosition() const; 95 96 /*Le confirma a entity el request de movimiento para comenzar la interpolaci 97 on*/ 98 virtual void move(Coordinate newPosition); 99 100 /*Por default no hace nada, las clases hijas deben reimplementarlo de ser 101 * necesario (por ejemplo el Priest)*/ 102 virtual void requestHeal(Player& player); 103 104 /*Actualiza el estado de la entity, por ejemplo si se esta moviendo le 105 actualiza la interpolacion*/ 106 void update(double timeStep, Game& game); 107 108 /*Retorna si el entity esta o no en movimiento (lo uso en monster)*/ 109 bool isMoving() const; 110 111 /*Retorna el tipo de entity (tipo de monstruo, npc o si es un player)*/ 112 GameType::Entity getType() const; 113 114 /*Carga los datos generales del entity de acuerdo al protocolo, se utiliza 115 * para la info que se le envia a un cliente recién conectado*/ 116 virtual void operator>>(std::stringstream& buffer) const; 117 118 virtual ~Entity() = default; 119 120 /*Retorna el nickname del entity (los monsters/npc tiene asignado uno tambie 121 n*/ 122 virtual const std::string& getNickname() const; 123 124 /*Actualiza la interpolacion de la entity cuando se esta moviendo 125 * de un tile al otro*/ 126 int32_t executeDisplacement(int32_t displacement, bool& hasFinished); 127 128 /*Retorna la coordenada del tile al que se esta desplazando la entity*/ </pre>	<pre> 63 que tiene guardados, por default no hace nada, debe ser reimplementada 64 si la clase hija puede almacenar y devolver items*/ 65 virtual void withdraw(Player& player, const std::string& itemName); 66 67 /*Implementa el comportamiento realizado al pedirle que guarde el item que s 68 e 69 encuentra en la posicion pasada, por default no hace nada, debe ser reimplem 70 entada 71 si la clase hija puede almacenar y devolver items*/ 72 virtual void deposit(Player& player, const std::string& itemName); 73 74 /*Implementa el comportamiento realizado al intentar comprar un item 75 con el nombre pasado, por default no hace nada, debe ser reimplementada 76 si la clase hija puede comprar y vender items*/ 77 virtual void buy(Player& player, const std::string& itemName); 78 79 /*Implementa el comportamiento realizado al intentar vender un item que se 80 encuentra en la posicion pasada, por default no hace nada, debe ser reimplem 81 entada 82 si la clase hija puede comprar y vender items*/ 83 virtual void sell(Player& player, const std::string& itemName); 84 85 /*Indica si el Entity va a ser atacado por un monstruo, por default retorna 86 false, las clases que hereden deben reimplementarla si son consideradas 87 targets*/ 88 virtual bool isMonsterTarget(); 89 90 /*Le asigna al jugador la posicion recibida*/ 91 void setPosition(Coordinate coordinate); 92 93 /*Retorna la posicion en la que quiere estar el jugadoras*/ 94 Coordinate getPosition() const; 95 96 /*Le confirma a entity el request de movimiento para comenzar la interpolaci 97 on*/ 98 virtual void move(Coordinate newPosition); 99 100 /*Por default no hace nada, las clases hijas deben reimplementarlo de ser 101 * necesario (por ejemplo el Priest)*/ 102 virtual void requestHeal(Player& player); 103 104 /*Actualiza el estado de la entity, por ejemplo si se esta moviendo le 105 actualiza la interpolacion*/ 106 void update(double timeStep, Game& game); 107 108 /*Retorna si el entity esta o no en movimiento (lo uso en monster)*/ 109 bool isMoving() const; 110 111 /*Retorna el tipo de entity (tipo de monstruo, npc o si es un player)*/ 112 GameType::Entity getType() const; 113 114 /*Carga los datos generales del entity de acuerdo al protocolo, se utiliza 115 * para la info que se le envia a un cliente recién conectado*/ 116 virtual void operator>>(std::stringstream& buffer) const; 117 118 virtual ~Entity() = default; 119 120 /*Retorna el nickname del entity (los monsters/npc tiene asignado uno tambie 121 n*/ 122 virtual const std::string& getNickname() const; 123 124 /*Actualiza la interpolacion de la entity cuando se esta moviendo 125 * de un tile al otro*/ 126 int32_t executeDisplacement(int32_t displacement, bool& hasFinished); 127 128 /*Retorna la coordenada del tile al que se esta desplazando la entity*/ </pre>

jul 21, 20 15:47

Entity.h

Page 3/3

```

124     virtual Coordinate getFinalCoordinate(GameType::Direction direction);
125
126     virtual int32_t getLevel() const;
127 };
128
129
130 #endif // ARGENTUM_ENTITY_H

```

jul 21, 20 15:47

Entity.cpp

Page 1/4

```

1  //
2  // Created by agustin on 8/6/20.
3  //
4
5  #include "Entity.h"
6  #include "AttackResult.h"
7  #include "../Game/Game.h"
8  #include "../Game/Events/Moved.h"
9  #include <msgpack.hpp>
10 #include "../libs/SharedConstants.h"
11
12 MSGPACK_ADD_ENUM(GameType::EventID)
13 MSGPACK_ADD_ENUM(GameType::Entity)
14 MSGPACK_ADD_ENUM(GameType::Direction)
15
16 unsigned int Entity::availableId = 0;
17
18 Entity::Entity(GameType::Entity _type, Coordinate initialPosition,
19               const std::string& _nicknamePrefix, bool isPrefixUnique /*= false*/) {
20     currentPosition.iPosition = initialPosition.iPosition;
21     currentPosition.jPosition = initialPosition.jPosition;
22     movement.movedDistance = 0;
23     movement.isMoving = false;
24     movement.direction = GameType::DIRECTION_STILL;
25     type = _type;
26     nickname = std::move(_nicknamePrefix);
27     if (!isPrefixUnique) {
28         nickname += std::to_string(availableId);
29         availableId++;
30     }
31 }
32
33 void Entity::setPosition(Coordinate coordinate) {
34     currentPosition = coordinate;
35 }
36
37 Coordinate Entity::getPosition() const {
38     return currentPosition;
39 }
40
41 bool Entity::isMonsterTarget() {
42     return false;
43 }
44
45 AttackResult Entity::attacked(int damage, unsigned int level, bool isAPlayer) {
46     return {0, 0};
47 }
48
49 void Entity::list(Player &player) {
50     //DO NOTHING
51 }
52
53 void Entity::withdraw(Player &player, const std::string& itemName) {
54     //DO NOTHING
55 }
56
57 void Entity::deposit(Player &player, const std::string& itemName) {
58     //DO NOTHING
59 }
60
61 void Entity::buy(Player &player, const std::string& itemName) {
62     //DO NOTHING
63 }
64
65 void Entity::sell(Player &player, const std::string& itemName) {
66     //DO NOTHING

```

jul 21, 20 15:47

Entity.cpp

Page 2/4

```

67 }
68
69
70 void Entity::requestHeal(Player &player) {
71     //DO NOTHING
72 }
73
74 void Entity::move(Coordinate newPosition) {
75     movement.direction = _getMoveDirection(newPosition);
76     currentPosition = newPosition;
77     movement.isMoving = true;
78 }
79
80 void Entity::update(double timeStep, Game& game) {
81     if (movement.isMoving) {
82         std::unique_ptr<Moved> event(new Moved(*this, movement.direction,
83             static_cast<unsigned int>(timeStep) * spe
84 ed));
85         game.pushEvent(std::move(event));
86     }
87
88 bool Entity::isMoving() const {
89     return movement.isMoving;
90 }
91
92 int32_t Entity::attack(Coordinate target) {
93     return -1;
94 }
95
96 GameType::Entity Entity::getType() const {
97     return type;
98 }
99
100 void Entity::operator>>(std::stringstream& buffer) const {
101     msgpack::type::tuple<GameType::EventID> idType(GameType::EventID::CREATE_ENT
102 ITY);
103     msgpack::pack(buffer, idType);
104     msgpack::type::tuple<GameType::Entity, std::string, int32_t>
105         idData(type, nickname, getLevel());
106     msgpack::pack(buffer, idData);
107
108     Coordinate previousPosition = _calculatePreviousPosition();
109     msgpack::type::tuple<int32_t, int32_t, GameType::Direction, int32_t> current
110 MovementData(previousPosition.iPosition,
111     previousPosition.jPosition, movement.direction, movement.movedDistanc
112 ce);
113     msgpack::pack(buffer, currentMovementData);
114 }
115
116 const std::string &Entity::getNickname() const {
117     return nickname;
118 }
119
120 int32_t Entity::executeDisplacement(int32_t displacement, bool& hasFinished) {
121     int32_t realDisplacement = displacement;
122     movement.movedDistance += displacement;
123     hasFinished = false;
124     if (movement.movedDistance ≥ TILE_DISTANCE_IN_METERS) {
125         realDisplacement = displacement - (movement.movedDistance - TILE_DISTANC
126 E_IN_METERS);
127         movement.movedDistance = 0;
128         movement.direction = GameType::DIRECTION_STILL;
129         movement.isMoving = false;
130         hasFinished = true;

```

jul 21, 20 15:47

Entity.cpp

Page 3/4

```

128 }
129     return realDisplacement;
130 }
131
132 Coordinate Entity::getFinalCoordinate(GameType::Direction moveDirection) {
133     if (!isMoving()) {
134         switch (moveDirection) {
135             case GameType::DIRECTION_UP:
136                 return {currentPosition.iPosition - 1, currentPosition.jPosition};
137             case GameType::DIRECTION_DOWN:
138                 return {currentPosition.iPosition + 1, currentPosition.jPosition};
139             case GameType::DIRECTION_RIGHT:
140                 return {currentPosition.iPosition, currentPosition.jPosition + 1};
141             case GameType::DIRECTION_LEFT:
142                 return {currentPosition.iPosition, currentPosition.jPosition - 1};
143             case GameType::DIRECTION_STILL:
144                 //do nothing
145                 break;
146         }
147     }
148     return {-1, -1};
149 }
150
151 int32_t Entity::getLevel() const {
152     return -1;
153 }
154
155 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
156 //Calcula la posicion previa tomando en cuenta la direccion de movimiento y la
157 //posicion actual
158 Coordinate Entity::calculatePreviousPosition() const {
159     Coordinate previous = currentPosition;
160     switch (movement.direction) {
161         case GameType::DIRECTION_UP:
162             previous.iPosition++;
163             break;
164         case GameType::DIRECTION_DOWN:
165             previous.iPosition--;
166             break;
167         case GameType::DIRECTION_LEFT:
168             previous.jPosition++;
169             break;
170         case GameType::DIRECTION_RIGHT:
171             previous.jPosition--;
172             break;
173         case GameType::DIRECTION_STILL:
174             //DO NOTHING
175             break;
176     }
177     return previous;
178 }
179
180 ///////////////////////////////////////////////////PROTECTED////////////////////////////////////
181 GameType::Direction Entity::getMoveDirection(Coordinate destination) {
182     Coordinate difference = {destination.iPosition - currentPosition.iPosition,
183         destination.jPosition - currentPosition.jPosition};
184     if (difference.iPosition == 1) {
185         return GameType::DIRECTION_DOWN;
186     } else if (difference.iPosition == -1) {
187         return GameType::DIRECTION_UP;
188     } else if (difference.jPosition == 1) {
189         return GameType::DIRECTION_LEFT;
190     } else if (difference.jPosition == -1) {
191         return GameType::DIRECTION_RIGHT;
192     }
193     return GameType::DIRECTION_STILL;

```

jul 21, 20 15:47

Entity.cpp

Page 4/4

```

190         return GameType::DIRECTION_LEFT;
191     } else {
192         return GameType::DIRECTION_RIGHT;
193     }
194 }
195

```

jul 21, 20 15:47

Trader.h

Page 1/1

```

1  //
2  // Created by agustin on 15/6/20.
3  //
4
5  #ifndef ARGENTUM_TRADER_H
6  #define ARGENTUM_TRADER_H
7
8
9  #include "../Entity.h"
10 #include "Shop.h"
11
12 class EntityTests;
13
14 class Trader: public Entity {
15 private:
16     Shop shop;
17
18     friend EntityTests;
19
20 public:
21     explicit Trader(Coordinate initialPosition);
22
23     /*Retorna por le minichat del player los items que tiene el Priest a la vent
24     a junto con su precio*/
25     void list(Player &player) override;
26
27     /*Le vende al player el item pedido en caso de tenerlo, caso contrario no su
28     cede nada*/
29     void buy(Player& player, const std::string& itemName) override;
30
31     /*Le compra al player el item pedido en caso de tenerlo, caso contrario no s
32     uce nada*/
33     void sell(Player& player, const std::string& itemName) override;
34 };
35
36 #endif // ARGENTUM_TRADER_H

```


jul 21, 20 15:47

Trader.cpp

Page 1/2

```

1  //
2  // Created by agustin on 15/6/20.
3  //
4
5  #include "Trader.h"
6  #include "../Config/Configuration.h"
7
8  #define INITIAL_ITEMS_AMOUNT 10
9  #define BUYING_PRICE_MULTIPLIER 1.1
10 #define SELLING_PRICE_MULTIPLIER 0.9
11
12 using namespace GameType;
13
14 Trader::Trader(Coordinate initialPosition) : Entity(GameType::TRADER, initialPos
15 ition, "Trader") {
16     std::unordered_set<std::string> acceptedProducts;
17     Configuration& config = Configuration::getInstance();
18     std::unordered_map<std::string, unsigned int> initialItemsAmounts;
19
20     initialItemsAmounts[config.configWeaponData(LONGSWORD).name] = INITIAL_ITEMS
21 _AMOUNT;
22     initialItemsAmounts[config.configWeaponData(AXE).name] = INITIAL_ITEMS_AMOUN
23 T;
24     initialItemsAmounts[config.configWeaponData(WARHAMMER).name] = INITIAL_ITEMS
25 _AMOUNT;
26     initialItemsAmounts[config.configWeaponData(SIMPLE_BOW).name] = INITIAL_ITEM
27 S_AMOUNT;
28     initialItemsAmounts[config.configWeaponData(COMPOSITE_BOW).name] = INITIAL_I
29 TEMS_AMOUNT;
30
31     initialItemsAmounts[config.configClothingData(LEATHER_ARMOR).name] = INITIAL
32 _ITEMS_AMOUNT;
33     initialItemsAmounts[config.configClothingData(PLATE_ARMOR).name] = INITIAL_I
34 TEMS_AMOUNT;
35     initialItemsAmounts[config.configClothingData(KING_ARMOR).name] = INITIAL_IT
36 EMS_AMOUNT;
37     initialItemsAmounts[config.configClothingData(BLUE_TUNIC).name] = INITIAL_IT
38 EMS_AMOUNT;
39     initialItemsAmounts[config.configClothingData(HOOD).name] = INITIAL_ITEMS_AM
40 OUNT;
41     initialItemsAmounts[config.configClothingData(IRON_HELMET).name] = INITIAL_I
42 TEMS_AMOUNT;
43     initialItemsAmounts[config.configClothingData(TURTLE_SHIELD).name] = INITIAL
44 _ITEMS_AMOUNT;
45     initialItemsAmounts[config.configClothingData(IRON_SHIELD).name] = INITIAL_I
46 TEMS_AMOUNT;
47     initialItemsAmounts[config.configClothingData(MAGIC_HAT).name] = INITIAL_ITE
48 MS_AMOUNT;
49
50     initialItemsAmounts[config.configPotionData(MANA_POTION).name] = INITIAL_ITE
51 MS_AMOUNT;
52     initialItemsAmounts[config.configPotionData(HEALTH_POTION).name] = INITIAL_I
53 TEMS_AMOUNT;
54
55     for (const auto & item: initialItemsAmounts) {
56         acceptedProducts.emplace(item.first);
57     }
58
59     Shop aux(initialItemsAmounts, std::move(acceptedProducts), BUYING_PRICE_MULT
60 IPLIER, SELLING_PRICE_MULTIPLIER);
61     shop = std::move(aux);
62 }
63
64 void Trader::list(Player &player) {
65     shop.list(player);
66 }

```

jul 21, 20 15:47

Trader.cpp

Page 2/2

```

49
50
51 void Trader::buy(Player &player, const std::string &itemName) {
52     shop.buy(player, itemName);
53 }
54
55 void Trader::sell(Player &player, const std::string& itemName) {
56     shop.sell(player, itemName);
57 }
58

```

jul 21, 20 15:47	Storage.h	Page 1/2
<pre> 1 // 2 // Created by agustin on 15/6/20. 3 // 4 5 #ifndef ARGENTUM_STORAGE_H 6 #define ARGENTUM_STORAGE_H 7 8 #include <string> 9 #include <unordered_map> 10 #include <list> 11 #include <utility> 12 #include <memory> 13 #include "../Server/PlayerData.hpp" 14 15 class Item; 16 class Player; 17 18 class EntityTests; 19 20 /*Clase utilizada para guardar instancias de items, se guardan las instancias 21 y no cantidades de cada item ya que de esa forma no se debe estar creando y 22 copiando cada vez que se retira algun item. Esto permite ademas extensibilidad 23 ya que se podria agregar a cada item por separado modificadores y estos quedaria 24 n 25 guardados en la instancia*/ 26 27 /*Cualquier tipo de item que sea guardado 0 veces no tiene un lugar en el unorde 28 red_map*/ 29 30 class Storage { 31 private: 32 int32_t storedGold; 33 std::unordered_map<std::string, std::list<std::shared_ptr<Item>>> storedItem 34 s; 35 36 friend EntityTests; 37 38 private: 39 static void _addAmmountMessageToPlayer(Player& player, const std::string& it 40 emName, 41 42 int concatenatedNumber); 43 44 public: 45 Storage(); 46 47 Storage(Storage^ other) noexcept; 48 49 Storage& operator=(Storage^ other) noexcept; 50 51 explicit Storage(const std::unordered_map<std::string, unsigned int>& 52 initialItemsAmounts, unsigned int gold); 53 54 /*Almacena el item, apropiandose de el. 55 Retorna true si cambio la cantidad de items guardados, sino retorna false 56 La cantidad cambia si se recibe algo que no sea oro*/ 57 bool storeItem(std::shared_ptr<Item>^ item); 58 59 /*Intenta pasar el item pedido de Storage a Player, si el item no existe o 60 el player no tiene espacio entonces no hace nada 61 Retorna true si cambio la cantidad de items que guarda, sino retorna false 62 La cantidad cambia si se saca algo que no sea oro*/ 63 bool retrieveItem(const std::string& itemName, Player& player); 64 65 /*Agrega en el minichat del player la lista con los items disponibles y su 66 * precio, la usan los vendedores*/ 67 void getStorageData(Player& player, const std::unordered_map<std::string,</pre>	<pre> 63 unsigned int>& prices, float priceMultiplier) const; 64 65 /*Agrega en el minichat del player la lista con los items disponibles, 66 * la usa el banker*/ 67 void getStorageData(Player& player) const; 68 69 /*Indica si el item con el nombre indicado se encuentra guardado*/ 70 bool isItemAvailable(const std::string& itemName) const; 71 72 /*Aumenta la cantidad de oro guardado en el storage*/ 73 void increaseGoldReserves(int amount); 74 75 /*Disminuye la cantidad de oro guardado en el storage*/ 76 bool decreaseGoldReserves(int amount); 77 78 /*Retorna en el struct playerData los datos de los items que el player 79 * tiene guardados*/ 80 void getPlayerData(PlayerData& playerData) const; 81 }; 82 83 #endif //ARGENTUM_STORAGE_H</pre>	<pre> 63 unsigned int>& prices, float priceMultiplier) const; 64 65 /*Agrega en el minichat del player la lista con los items disponibles, 66 * la usa el banker*/ 67 void getStorageData(Player& player) const; 68 69 /*Indica si el item con el nombre indicado se encuentra guardado*/ 70 bool isItemAvailable(const std::string& itemName) const; 71 72 /*Aumenta la cantidad de oro guardado en el storage*/ 73 void increaseGoldReserves(int amount); 74 75 /*Disminuye la cantidad de oro guardado en el storage*/ 76 bool decreaseGoldReserves(int amount); 77 78 /*Retorna en el struct playerData los datos de los items que el player 79 * tiene guardados*/ 80 void getPlayerData(PlayerData& playerData) const; 81 }; 82 83 #endif //ARGENTUM_STORAGE_H</pre>

jul 21, 20 15:47

Storage.cpp

Page 1/3

```

1  //
2  // Created by agustin on 15/6/20.
3  //
4
5  #include "Storage.h"
6  #include "../Items/ItemsFactory.h"
7  #include <utility>
8  #include "../Items/Item.h"
9  #include "../Player.h"
10 #include "msgpack.hpp"
11 #include "../Config/Configuration.h"
12
13 MSGPACK_ADD_ENUM(GameType::EventID)
14
15 #define UNEXISTING_ITEM_MESSAGE "The requested item is not available\n"
16
17 Storage &Storage::operator=(Storage &other) noexcept {
18     storedGold = other.storedGold;
19     other.storedGold = 0;
20     storedItems = std::move(other.storedItems);
21     return *this;
22 }
23
24 Storage::Storage(Storage &other) noexcept {
25     storedGold = other.storedGold;
26     other.storedGold = 0;
27     storedItems = std::move(other.storedItems);
28 }
29
30 Storage::Storage(const std::unordered_map<std::string, unsigned int>&
31     initialItemsAmounts, unsigned int gold) {
32     storedGold = gold;
33     ItemsFactory& factory = ItemsFactory::getInstance();
34     std::shared_ptr<Item> aux;
35     for (const auto & initialItemAmount: initialItemsAmounts) {
36         for (unsigned int i = 0; i < initialItemAmount.second; ++i) {
37             factory.storeItemInstance(initialItemAmount.first, aux);
38             storedItems[initialItemAmount.first].push_back(std::move(aux));
39         }
40     }
41 }
42
43 bool Storage::storeItem(std::shared_ptr<Item> &item) {
44     if (item) {
45         storedItems[item->getName()].push_back(std::move(item));
46         return true;
47     }
48     return false;
49 }
50
51 bool Storage::retrieveItem(const std::string& itemName, Player &player) {
52     std::shared_ptr<Item> item;
53     if (storedItems.count(itemName) == 1) {
54         item = storedItems.at(itemName).front();
55         if (!player.storeItem(item)) {
56             return false;
57         }
58         storedItems[itemName].pop_front();
59         if (storedItems[itemName].empty()) {
60             storedItems.erase(itemName);
61         }
62     } else {
63         player.addMessage(UNEXISTING_ITEM_MESSAGE);
64         return false;
65     }
66     return true;

```

jul 21, 20 15:47

Storage.cpp

Page 2/3

```

67 }
68
69 void Storage::getStorageData(Player& player, const std::unordered_map<std::string,
70     unsigned int> &prices, float priceMultiplier) const {
71     _addAmountMessageToPlayer(player, Configuration::getInstance().configGetGoldName(),
72         storedGold);
73     for (const auto & storedItem : storedItems) {
74         _addAmountMessageToPlayer(player, storedItem.second.front()->getName(),
75             prices.at(storedItem.first) * priceMultiplier);
76     }
77 }
78
79 void Storage::getStorageData(Player& player) const {
80     _addAmountMessageToPlayer(player, Configuration::getInstance().configGetGoldName(),
81         storedGold);
82     for (const auto & storedItem : storedItems) {
83         _addAmountMessageToPlayer(player, storedItem.second.front()->getName(),
84             storedItem.second.size());
85     }
86 }
87
88 bool Storage::isItemAvailable(const std::string &itemName) const {
89     return storedItems.count(itemName) == 1;
90 }
91
92 void Storage::increaseGoldReserves(int amount) {
93     storedGold += amount;
94 }
95
96 bool Storage::decreaseGoldReserves(int amount) {
97     if (amount <= storedGold) {
98         storedGold -= amount;
99         return true;
100     }
101     return false;
102 }
103
104 Storage::Storage() {
105     storedGold = 0;
106 }
107
108 void Storage::getPlayerData(PlayerData &playerData) const {
109     int stored = 0;
110     for (auto & item : storedItems) {
111         GameType::ItemType type = item.second.front()->getType();
112         int32_t id = item.second.front()->getId();
113         for (std::size_t i = 0; i < item.second.size(); ++i) {
114             playerData.bankerItems.at(stored) = std::make_tuple(type, id);
115             ++stored;
116         }
117     }
118     playerData.bankerGold = storedGold;
119 }
120
121 ///////////////////////////////////////////////////PRIVATE////////////////////////////////////
122
123 void Storage::_addAmountMessageToPlayer(Player &player, const std::string &itemName,
124     int concatenatedNumber) {
125     player.addMessage(itemName);
126     player.addMessage(": ");
127     player.addMessage(std::to_string(concatenatedNumber));
128     player.addMessage("\n");

```

jul 21, 20 15:47

Storage.cpp

Page 3/3

128 }

jul 21, 20 15:47

Shop.h

Page 1/1

```

1  //
2  // Created by agustin on 16/6/20.
3  //
4
5  #ifndef ARGENTUM_SHOP_H
6  #define ARGENTUM_SHOP_H
7
8  #include "Storage.h"
9
10 class Player;
11
12 class EntityTests;
13
14
15 class Shop {
16 private:
17     float buyingMultiplier{};
18     float sellingMultiplier{};
19     Storage storage;
20     std::unordered_map<std::string, unsigned int> prices;
21     std::unordered_set<std::string> acceptedProducts;
22
23     friend EntityTests;
24
25 public:
26     Shop();
27
28     Shop(const std::unordered_map<std::string, unsigned int>&
29         initialItemsAmounts, std::unordered_set<std::string>& acceptedProducts
30         , float buyingMultiplier, float sellingMultiplier)
31     ;
32
33     Shop(Shop& other) noexcept;
34
35     Shop& operator=(Shop& other) noexcept;
36
37     /*Almacena en la lista los items que tiene en venta, los precios
38     almacenados para cada producto fueron modificados utilizando el
39     multiplicador recibido en el constructor*/
40     void list(Player &player) const;
41
42     /*Funcion utilizada cuando el jugador quiere comprar un item, si el jugador
43     no tiene oro suficiente o el item pedido no esta guardado entonces no hace nada
44     */
45     void buy(Player& player, const std::string& itemName);
46
47     /*Funcion utilizada cuando el jugador quiere vender un item, si el NPC compr
48     ador no tiene oro suficiente o el item pedido no esta guardado entonces no hace nada
49     */
50     void sell(Player& player, const std::string& itemName);
51 };
52
53 #endif // ARGENTUM_SHOP_H

```

jul 21, 20 15:47

Shop.cpp

Page 1/2

```

1  //
2  // Created by agustin on 16/6/20.
3  //
4
5  #include "Shop.h"
6  #include "../Player.h"
7  #include "../Config/Configuration.h"
8
9
10 #define PRODUCT_NOT_IN_STORAGE_MESSAGE "I don't have a "
11 #define NOT_ACCEPTED_PRODUCT_MESSAGE "I don't buy "
12 #define NOT_ENOUGH_GOLD_STORED_MESSAGE "I don't have enough gold\n"
13 #define PLAYER_CANT_AFFORD_MESSAGE "You don't have enough gold\n"
14 #define PLAYER_DOES_NOT_HAVE_SPACE_MESSAGE "You don't have enough space for that item\n"
15
16 Shop::Shop() {
17     sellingMultiplier = 1;
18     buyingMultiplier = 1;
19 }
20
21 Shop::Shop(const std::unordered_map<std::string, unsigned int> &initialItemsAmounts,
22            std::unordered_set<std::string>^ _acceptedProducts, float _buyingMultiplier, float _sellingMultiplier):
23     storage(initialItemsAmounts, Configuration::getInstance().configInitialMerchantGold()) {
24     acceptedProducts = std::move(_acceptedProducts);
25     Configuration& config = Configuration::getInstance();
26     const auto & weaponsData = config.configAllWeaponsData();
27     const auto & clothesData = config.configAllClothingData();
28     const auto & potionsData = config.configAllPotionsData();
29
30     for (const auto & weaponData: weaponsData) {
31         prices[weaponData.second.name] = weaponData.second.price;
32     }
33     for (const auto & clothingData: clothesData) {
34         prices[clothingData.second.name] = clothingData.second.price;
35     }
36     for (const auto & potionData: potionsData) {
37         prices[potionData.second.name] = potionData.second.price;
38     }
39     buyingMultiplier = _buyingMultiplier;
40     sellingMultiplier = _sellingMultiplier;
41 }
42
43 Shop &Shop::operator=(Shop &other) noexcept {
44     storage = std::move(other.storage);
45     buyingMultiplier = other.buyingMultiplier;
46     sellingMultiplier = other.sellingMultiplier;
47     prices = std::move(other.prices);
48     acceptedProducts = std::move(other.acceptedProducts);
49     return *this;
50 }
51
52 Shop::Shop(Shop &other) noexcept {
53     storage = std::move(other.storage);
54     buyingMultiplier = other.buyingMultiplier;
55     sellingMultiplier = other.sellingMultiplier;
56     prices = std::move(other.prices);
57     acceptedProducts = std::move(other.acceptedProducts);
58 }
59
60 void Shop::list(Player &player) const {
61     storage.getStorageData(player, prices, buyingMultiplier);
62 }
63

```

jul 21, 20 15:47

Shop.cpp

Page 2/2

```

64 void Shop::buy(Player &player, const std::string &itemName) {
65     unsigned int price;
66     if (storage.isItemAvailable(itemName)) {
67         price = static_cast<unsigned int>(static_cast<float>(prices[itemName])
68                                         * buyingMultiplier);
69         if (!player.hasFullInventory()) {
70             if (player.spendGold(static_cast<int>(price))) {
71                 storage.increaseGoldReserves(static_cast<int>(price));
72                 storage.retrieveItem(itemName, player);
73             } else {
74                 player.addMessage(PLAYER_CANT_AFFORD_MESSAGE);
75             }
76         } else {
77             player.addMessage(PLAYER_DOES_NOT_HAVE_SPACE_MESSAGE);
78         }
79     } else {
80         player.addMessage(PRODUCT_NOT_IN_STORAGE_MESSAGE + itemName + "\n");
81     }
82 }
83
84 void Shop::sell(Player &player, const std::string& itemName) {
85     if (acceptedProducts.count(itemName) == 0) {
86         player.addMessage(NOT_ACCEPTED_PRODUCT_MESSAGE + itemName + "\n");
87         return;
88     }
89     unsigned int price;
90     price = static_cast<unsigned int>(static_cast<float>(prices.at(itemName))
91                                     * sellingMultiplier);
92     if (player.hasItem(itemName)) {
93         if (storage.decreaseGoldReserves(static_cast<int>(price))) {
94             player.receiveGold(price);
95             storage.storeItem(player.removeItem(itemName));
96             player.addMessage("You got " + std::to_string(price) + " gold coins\n");
97         } else {
98             player.addMessage(NOT_ENOUGH_GOLD_STORED_MESSAGE);
99         }
100     }
101 }

```

jul 21, 20 15:47

Priest.h

Page 1/1

```

1  //
2  // Created by agustin on 15/6/20.
3  //
4
5  #ifndef ARGENTUM_PRIEST_H
6  #define ARGENTUM_PRIEST_H
7
8
9  #include "Shop.h"
10 #include "../Entity.h"
11
12 class EntityTests;
13
14 class Priest: public Entity {
15 private:
16     Shop shop;
17
18     friend EntityTests;
19
20 public:
21     explicit Priest(Coordinate initialPosition);
22
23     /*Retorna por le minichat del player los items que tiene el Priest a la vent
a junto con su precio*/
24     void list(Player &player) override;
25
26     /*Le vende al player el item pedido en caso de tenerlo, caso contrario no su
cede nada*/
27     void buy(Player& player, const std::string& itemName) override;
28
29     /*Le compra al player el item pedido en caso de tenerlo, caso contrario no s
ucede nada*/
30     void sell(Player& player, const std::string& itemName) override;
31
32     /*Cura al Player toda su vida y le recupera la totalidad del mana tambien (n
o le carga mana al Warrior)*/
33     void requestHeal(Player& player) override;
34 };
35
36
37 #endif //ARGENTUM_PRIEST_H

```

jul 21, 20 15:47

Priest.cpp

Page 1/1

```

1  //
2  // Created by agustin on 15/6/20.
3  //
4
5  #include "Priest.h"
6  #include "../Config/Configuration.h"
7  #include "../Player.h"
8
9  using namespace GameType;
10
11 #define INITIAL_WEAPONS_AMOUNT 10
12 #define INITIAL_POTIONS_AMOUNT 40
13 #define BUYING_PRICE_MULTIPLIER 1.25
14 #define SELLING_PRICE_MULTIPLIER 0.75
15
16 Priest::Priest(Coordinate initialPosition): Entity(GameType::PRIEST, initialPosi
tion, "Priest") {
17     std::unordered_set<std::string> acceptedProducts;
18     Configuration& config = Configuration::getInstance();
19     std::unordered_map<std::string, unsigned int> initialItemsAmounts;
20     initialItemsAmounts[config.configWeaponData(LINKED_STAFF).name] = INITIAL_WE
APONS_AMOUNT;
21     initialItemsAmounts[config.configWeaponData(GNARLED_STAFF).name] = INITIAL_W
EAPONS_AMOUNT;
22     initialItemsAmounts[config.configWeaponData(ELVEN_FLUTE).name] = INITIAL_WEA
PONS_AMOUNT;
23     initialItemsAmounts[config.configWeaponData(ASH_ROD).name] = INITIAL_WEAPONS
_AMOUNT;
24
25     initialItemsAmounts[config.configPotionData(HEALTH_POTION).name] = INITIAL_P
OTIONS_AMOUNT;
26     initialItemsAmounts[config.configPotionData(MANA_POTION).name] = INITIAL_POT
IONS_AMOUNT;
27
28     for (const auto & item: initialItemsAmounts) {
29         acceptedProducts.emplace(item.first);
30     }
31
32     Shop aux(initialItemsAmounts, std::move(acceptedProducts), BUYING_PRICE_MULT
IPLIER, SELLING_PRICE_MULTIPLIER);
33     shop = std::move(aux);
34 }
35
36 void Priest::list(Player &player) {
37     shop.list(player);
38 }
39
40 void Priest::buy(Player &player, const std::string &itemName) {
41     shop.buy(player, itemName);
42 }
43
44 void Priest::sell(Player &player, const std::string& itemName) {
45     shop.sell(player, itemName);
46 }
47
48 void Priest::requestHeal(Player &player) {
49     player.restoreStats(false);
50 }
51

```

jul 21, 20 15:47

CitizenFactory.h

Page 1/1

```

1 //
2 // Created by agustin on 25/6/20.
3 //
4
5 #ifndef ARGENTUM_CITIZENFACTORY_H
6 #define ARGENTUM_CITIZENFACTORY_H
7
8 #include <unordered_map>
9 #include <memory>
10 #include "../libs/GameEnums.h"
11 #include "../Map/Coordinate.h"
12
13 class Entity;
14
15 typedef void (*CitizenCreator)(std::shared_ptr<Entity>& citizen, Coordinate initialPosition);
16
17 /*Factory de citizens*/
18
19 class CitizenFactory {
20 private:
21     std::unordered_map<GameType::Entity, CitizenCreator> citizensCreators;
22
23 private:
24     static void _storeTrader(std::shared_ptr<Entity>& citizen, Coordinate initialPosition);
25     static void _storePriest(std::shared_ptr<Entity>& citizen, Coordinate initialPosition);
26     static void _storeBanker(std::shared_ptr<Entity>& citizen, Coordinate initialPosition);
27
28 public:
29     CitizenFactory();
30
31     /*Crea un citizen y lo almacena en el shared_ptr*/
32     void storeCitizen(std::shared_ptr<Entity>& citizen, GameType::Entity _type, Coordinate initialPosition);
33 };
34
35
36 #endif //ARGENTUM_CITIZENFACTORY_H

```

jul 21, 20 15:47

CitizenFactory.cpp

Page 1/1

```

1 //
2 // Created by agustin on 25/6/20.
3 //
4
5 #include "CitizenFactory.h"
6 #include "Trader.h"
7 #include "Priest.h"
8 #include "Banker.h"
9
10 //////////////////////////////////PRIVATE////////////////////////////////////
11
12 void CitizenFactory::_storeTrader(std::shared_ptr<Entity>& citizen, Coordinate initialPosition) {
13     citizen = std::make_shared<Trader>(initialPosition);
14 }
15
16
17 void CitizenFactory::_storePriest(std::shared_ptr<Entity>& citizen, Coordinate initialPosition) {
18     citizen = std::make_shared<Priest>(initialPosition);
19 }
20
21 void CitizenFactory::_storeBanker(std::shared_ptr<Entity> &citizen, Coordinate initialPosition) {
22     citizen = std::make_shared<Banker>(initialPosition);
23 }
24
25 //////////////////////////////////PUBLIC////////////////////////////////////
26
27 CitizenFactory::CitizenFactory() {
28     citizensCreators[GameType::Entity::TRADER] = _storeTrader;
29     citizensCreators[GameType::Entity::PRIEST] = _storePriest;
30     citizensCreators[GameType::Entity::BANKER] = _storeBanker;
31 }
32
33 void CitizenFactory::storeCitizen(std::shared_ptr<Entity> &citizen, GameType::Entity _type, Coordinate initialPosition) {
34     citizensCreators.at(_type)(citizen, initialPosition);
35 }
36

```


jul 21, 20 15:47

Banker.cpp

Page 1/3

```

1  //
2  // Created by agustin on 15/6/20.
3  //
4
5  #include "Banker.h"
6
7  #include "../Player.h"
8  #include "Storage.h"
9  #include "../libs/TPEException.h"
10 #include "../Config/Configuration.h"
11 #include "../Server/NonModifiableConstants.h"
12
13 #define NO_ROOM_AVAILABLE_MESSAGE "You don't have more storage room, the limit is "
14 #define ROOM_AVAILABLE_MESSAGE "Items stored: "
15 #define NO_ITEM_MESSAGE "You don't have that item in you inventory\n"
16 #define INVALID_GOLD_PARAMETERS "Invalid parameters for gold deposit/withdrawal\n"
17 #define INSUFFICIENT_GOLD_MESSAGE "Insufficient gold\n"
18 #define NEGATIVE_GOLD_MESSAGE "Negative gold does not exist\n"
19 #define GOLD_AMOUNT_SEPARATOR ' '
20
21 std::unordered_map<std::string, std::pair<unsigned int, Storage>> Banker::player
sStorages;
22
23 Banker::Banker(Coordinate initialPosition): Entity(GameType::BANKER,
24             initialPosition, "Banker") {
25
26 }
27
28 void Banker::list(Player &player) {
29     const std::pair<unsigned int, Storage>& aux = playersStorages.at(player.getN
ickname());
30     aux.second.getStorageData(player);
31     _storeAvailableRoomMessage(player, aux.first);
32 }
33
34 void Banker::withdraw(Player &player, const std::string &itemName) {
35     try {
36         std::pair<unsigned int, Storage>& aux = playersStorages.at(player.getNic
kname());
37         if (itemName.find(Configuration::getInstance().configGetGoldName()) != st
d::string::npos) {
38             _modifyGoldReserves(aux.second, player, itemName, _withdrawGold);
39         } else if (aux.second.retrieveItem(itemName, player)) {
40             aux.first--;
41             _storeAvailableRoomMessage(player, aux.first);
42         }
43     } catch (...) {
44         throw TPEException("Tried to withdraw an item from a player that "
45             "didn't exist!");
46     }
47 }
48
49 void Banker::deposit(Player &player, const std::string& itemName) {
50     try {
51         std::pair<unsigned int, Storage>& aux = playersStorages.at(player.getNic
kname());
52         if (itemName.find(Configuration::getInstance().configGetGoldName()) != st
d::string::npos) {
53             _modifyGoldReserves(aux.second, player, itemName, _depositGold);
54         } else if (aux.first < BANK_SIZE) {
55             if (playersStorages.at(player.getNickname()).second.storeItem(player
.removeItem(itemName))) {
56                 aux.first++;
57                 _storeAvailableRoomMessage(player, aux.first);
58             } else {
59

```

jul 21, 20 15:47

Banker.cpp

Page 2/3

```

60         player.addMessage(NO_ITEM_MESSAGE);
61     }
62     } else {
63         player.addMessage(NO_ROOM_AVAILABLE_MESSAGE + std::to_string(BANK_SI
ZE) + "\n");
64     }
65     } catch (...) {
66         throw TPEException("Tried to deposit an item of an inexistent player!");
67     }
68 }
69
70 void Banker::getPlayerItems(PlayerData &playerData) {
71     auto & playerStorageData = playersStorages.at(playerData.nickname);
72     playerStorageData.second.getPlayerData(playerData);
73 }
74
75 void Banker::addPlayerItems(const PlayerData &playerData) {
76     std::unordered_map<std::string, unsigned int> initialItemsAmounts;
77     unsigned int itemAmount = 0;
78     std::tuple<GameType::ItemType, int32_t> currItemType = playerData.bankerItem
s.at(0);
79     for (auto & item : playerData.bankerItems) {
80         if (currItemType != item) {
81             std::string itemName = _translateItemTypeToName(currItemType);
82             initialItemsAmounts.emplace(itemName, itemAmount);
83             itemAmount = 0;
84             currItemType = item;
85         }
86         ++itemAmount;
87     }
88     unsigned int gold = playerData.bankerGold;
89     playersStorages.emplace(playerData.nickname, std::pair<int32_t, Storage>
90         (_getNumberOfItemsStored(initialItemsAmounts), Storage(initialItemsA
mounts, gold)));
91 }
92
93
94 void Banker::erasePlayerItems(const std::string& playerNickname) {
95     playersStorages.erase(playerNickname);
96 }
97
98
99
100 //////////////////////////////////PRIVATE////////////////////////////////////
101
102 int32_t Banker::_getNumberOfItemsStored(const std::unordered_map<std::string, un
signed int> &
103             initialItemsAmounts) {
104     int32_t storedItemsAmount = 0;
105     for (const auto & itemList: initialItemsAmounts) {
106         storedItemsAmount += itemList.second;
107     }
108     return storedItemsAmount;
109 }
110
111 void Banker::_storeAvailableRoomMessage(Player &player, unsigned int storedItems
Amount) {
112     player.addMessage(ROOM_AVAILABLE_MESSAGE + std::to_string(storedItemsAmount)
113         + "/"
114         + std::to_string(BANK_SIZE) + "\n");
115 }
116 void Banker::_modifyGoldReserves(Storage& playerStorage, Player &player,
117     const std::string& itemName, ModifyGold modifyGold) {
118     int goldAmount = 0;

```

jul 21, 20 15:47

Banker.cpp

Page 3/3

```

119     size_t separatorPosition = itemName.find(GOLD_AMOUNT_SEPARATOR);
120     if ((separatorPosition != std::string::npos) ^
121         (separatorPosition != itemName.size() - 1)) {
122         try {
123             goldAmount = std::stoi(itemName.substr(separatorPosition + 1));
124             modifyGold(playerStorage, player, goldAmount);
125         } catch (std::invalid_argument &e) {
126             player.addMessage(INVALID_GOLD_PARAMETERS);
127         } catch (std::out_of_range &e) {
128             player.addMessage(INVALID_GOLD_PARAMETERS);
129         }
130     } else {
131         player.addMessage(INVALID_GOLD_PARAMETERS);
132     }
133 }
134
135 void Banker::_depositGold(Storage &playerStorage, Player &player, int goldAmount
136 ) {
137     if (goldAmount ≥ 0) {
138         if (player.spendGold(goldAmount)) {
139             playerStorage.increaseGoldReserves(goldAmount);
140         } else {
141             player.addMessage(INSUFFICIENT_GOLD_MESSAGE);
142         }
143     } else {
144         player.addMessage(NEGATIVE_GOLD_MESSAGE);
145     }
146 }
147
148 void Banker::_withdrawGold(Storage &playerStorage, Player &player, int goldAmount
149 ) {
150     if (goldAmount ≥ 0) {
151         if (playerStorage.decreaseGoldReserves(goldAmount)) {
152             player.receiveGold(goldAmount);
153         } else {
154             player.addMessage(INSUFFICIENT_GOLD_MESSAGE);
155         }
156     } else {
157         player.addMessage(NEGATIVE_GOLD_MESSAGE);
158     }
159 }
160
161 std::string Banker::_translateItemTypeToName(std::tuple<GameType::ItemType, int>
162 2_t item) {
163     Configuration &config = Configuration::getInstance();
164     switch (std::get<0>(item)) {
165     case GameType::ITEM_TYPE_CLOTHING:
166         return config.configClothingData(
167             static_cast<GameType::Clothing>(std::get<1>(item))).name;
168     case GameType::ITEM_TYPE_WEAPON:
169         return config.configWeaponData(
170             static_cast<GameType::Weapon>(std::get<1>(item))).name;
171     case GameType::ITEM_TYPE_POTION:
172         return config.configPotionData(
173             static_cast<GameType::Potion>(std::get<1>(item))).name;
174     default:
175         //do nothing
176         throw TPEException("Player was storing invalid banker items!");
177     }
178 }

```

jul 21, 20 15:47

AttackResult.h

Page 1/1

```

1 //
2 // Created by marcos on 18/6/20.
3 //
4
5 #ifndef ARGENTUM_ATTACKRESULT_H
6 #define ARGENTUM_ATTACKRESULT_H
7
8 struct AttackResult {
9     int damage;
10     unsigned int experience;
11     std::string resultMessage;
12 };
13
14 #endif //ARGENTUM_ATTACKRESULT_H

```

jul 21, 20 15:47

MapFileReader.h

Page 1/1

```

1  //
2  // Created by marcos on 6/15/20.
3  //
4
5  #ifndef ARGENTUM_MAPFILEREADER_H
6  #define ARGENTUM_MAPFILEREADER_H
7
8  #include <fstream>
9  #include "json.hpp"
10 #include <unordered_map>
11
12 struct TileInfo {
13     std::string tileType;
14     std::string structureType;
15     std::string entityType;
16     bool isOccupable;
17     bool isFromCity;
18 };
19
20 struct MapSize {
21     unsigned int width;
22     unsigned int height;
23 };
24
25 class MapFileReader {
26 private:
27     nlohmann::json obj;
28     std::unordered_map<int, std::string> mapElements;
29     MapSize mapDimensions{};
30
31 public:
32     explicit MapFileReader(const std::string& path);
33
34     /*Retorna los datos del tile (si es ocupable, pertenece a una ciudad, tiene u
35 n NPC/estructura*/
36     TileInfo getTileInfo(unsigned int x, unsigned int y);
37
38     /*Retorna las dimensiones MxN del mapa*/
39     MapSize getMapDimensions() const;
40
41 private:
42     void _readMapSize();
43     void _readIDs();
44 };
45
46 #endif //ARGENTUM_MAPFILEREADER_H

```

jul 21, 20 15:47

MapFileReader.cpp

Page 1/2

```

1  //
2  // Created by marcos on 6/15/20.
3  //
4
5  #include "MapFileReader.h"
6  #include "../libs/TPEException.h"
7  #include <memory>
8
9  using json = nlohmann::json;
10
11 MapFileReader::MapFileReader(const std::string& path) {
12     mapDimensions.width = 0;
13     mapDimensions.height = 0;
14     std::ifstream file(path);
15     if (!file.is_open()) {
16         throw TPEException("Could not open Map File, check whether"
17                             " it exists or not!");
18     }
19     try {
20         file >> obj;
21     } catch (...) {
22         throw TPEException("Map file parsing failed!");
23     }
24     _readMapSize();
25     _readIDs();
26 }
27
28 void MapFileReader::_readMapSize() {
29     mapDimensions.width = obj["width"].get<int>();
30     mapDimensions.height = obj["height"].get<int>();
31 }
32
33 void MapFileReader::_readIDs() {
34     json& tilesets = obj["tilesets"];
35     mapElements.emplace(0, "Nothing");
36     for (auto & tileset : tilesets) {
37         std::string name = tileset["name"].get<std::string>();
38         int id = tileset["firstgid"].get<int>();
39         int tilecount = tileset["tilecount"].get<int>();
40         if (tilecount > 1) {
41             for (int i = 0; i < tilecount; ++i) {
42                 mapElements.emplace(id + i, name + std::to_string(i));
43             }
44         } else {
45             mapElements.emplace(id, name);
46         }
47     }
48 }
49
50 TileInfo MapFileReader::getTileInfo(unsigned int row, unsigned int column) {
51     json& layers = obj["layers"];
52     json& tileData = layers[0]["data"];
53     TileInfo tile;
54     tile.tileType = mapElements.at(tileData[row*mapDimensions.width + column].ge
55 t<int>());
56     json& sData = layers[1]["data"];
57     tile.structureType = mapElements.at(sData[row*mapDimensions.width + column].
58 get<int>());
59     json& eData = layers[2]["data"];
60     tile.entityType = mapElements.at(eData[row*mapDimensions.width + column].get
61 <int>());
62     json& oData = layers[3]["data"]; /*isOccupable*/
63     tile.isOccupable = (oData[row*mapDimensions.width + column].get<int>() == 0);
64     json& cData = layers[4]["data"]; /*isFromCity*/
65     tile.isFromCity = (cData[row*mapDimensions.width + column].get<int>() != 0);
66     return tile;
67 }

```

jul 21, 20 15:47

MapFileReader.cpp

Page 2/2

```

64 }
65
66 MapSize MapFileReader::getMapDimensions() const {
67     return mapDimensions;
68 }

```

jul 21, 20 15:47

Configuration.h

Page 1/3

```

1  //
2  // Created by ivan on 8/6/20.
3  //
4
5  #ifndef ARGENTUM_CONFIGURATION_H
6  #define ARGENTUM_CONFIGURATION_H
7
8  #include "ConfigFileReader.h"
9  #include <unordered_map>
10
11  /*La siguiente clase toma los valores que lee el ConfigFileReader y los guarda e
12   n memoria para que el resto del
13   * juego pueda acceder a ellos rapidamente*/
14
15  class Configuration {
16  private:
17      std::unordered_map<GameType::Race, Config::Modifiers> raceModifiers{};
18      std::unordered_map<GameType::Class, Config::Modifiers> classModifiers{};
19      std::unordered_map<GameType::Entity, Config::MonsterStats> monsterStats{};
20      std::unordered_map<GameType::Weapon, Config::WeaponData> weaponData{};
21      std::unordered_map<GameType::Clothing, Config::ClothingData> clothingData{};
22      std::unordered_map<GameType::Potion, Config::PotionData> potionData{};
23
24      Config::GoldModifiers goldModifiers{};
25
26      Config::XPModifiers xpModifiers{};
27
28      float criticalAttackChance;
29      float dodgeChance;
30
31      unsigned int newbieLevel;
32      unsigned int maxLevelDif;
33
34      unsigned int timeBetweenMonsterSpawns{};
35      unsigned int monsterSpawnAmount{};
36      unsigned int maxMonsterAmount{};
37
38      unsigned int initialMerchantGold;
39      std::string goldName;
40
41      unsigned int playerSpeed;
42      double timeForPlayerRecovery;
43
44      std::string port;
45      std::string mapPath;
46      std::string savePath;
47      std::string indexPath;
48
49  public:
50      Configuration(Configuration const&) = delete;
51      void operator=(Configuration const&) = delete;
52
53      /*Retorna la instancia Singleton del Configuration*/
54      static Configuration& getInstance();
55
56      /*Retorna los modificadores de clase*/
57      const Config::Modifiers& configClassModifiers(GameType::Class _class) const;
58
59      /*Retorna los modificadores de clase*/
60      const Config::Modifiers& configRaceModifiers(GameType::Race race) const;
61
62      /*Retorna los modificadores de clase*/
63      const Config::MonsterStats& configMonsterStats(GameType::Entity monster) con
64      st;

```

jul 21, 20 15:47	Configuration.h	Page 2/3
65	<i>/*Retorna los modificadores de clase*/</i>	
66	const Config::WeaponData& configWeaponData(GameType::Weapon weapon) const ;	
67		
68	<i>/*Retorna los modificadores de clase*/</i>	
69	const Config::ClothingData& configClothingData(GameType::Clothing clothes) const ;	
70		
71	<i>/*Retorna los modificadores de clase*/</i>	
72	const Config::PotionData& configPotionData(GameType::Potion potion) const ;	
73		
74	<i>/*Retorna los modificadores de clase*/</i>	
75	const Config::GoldModifiers& configGoldModifiers() const ;	
76		
77	<i>/*Retorna los modificadores de clase*/</i>	
78	const Config::XPModifiers& configXPModifiers() const ;	
79		
80	<i>/*Retorna los modificadores de clase*/</i>	
81	const std::unordered_map<GameType::Weapon, Config::WeaponData>& configAllWeaponsData();	
82		
83	<i>/*Retorna los modificadores de clase*/</i>	
84	const std::unordered_map<GameType::Clothing, Config::ClothingData>& configAllClothingData();	
85		
86	<i>/*Retorna los modificadores de clase*/</i>	
87	const std::unordered_map<GameType::Potion, Config::PotionData>& configAllPotionsData();	
88		
89	<i>/*Retorna la probabilidad de un ataque critico*/</i>	
90	float configCriticalAttackChance() const ;	
91		
92	<i>/*Retorna un coeficiente que afecta la probabilidad de esquivar un ataque*/</i>	
93	float configDodgeChance() const ;	
94		
95	<i>/*Retorna el newbie level*/</i>	
96	unsigned int configNewbieLevel() const ;	
97		
98	<i>/*Retorna la maxima diferencia que puede haber entre 2 players para que puedan atacarse*/</i>	
99	unsigned int configMaxLevelDif() const ;	
100		
101	<i>/*Retorna el tiempo entre spawns de monstruos*/</i>	
102	unsigned int configTimeBetweenMonsterSpawns() const ;	
103		
104	<i>/*Retorna la maxima cantidad de monstruos que puede haber en simultaneo vivos en la partida*/</i>	
105	unsigned int configMaxMonsterAmount() const ;	
106		
107	<i>/*Retorna la cantidad de monstruos que spawnen a la vez en cada spawn*/</i>	
108	unsigned int configMonsterSpawnAmount() const ;	
109		
110	<i>/*Retorna la cantidad inicial de oro de los mercaderes*/</i>	
111	unsigned int configInitialMerchantGold() const ;	
112		
113	<i>/*Retorna el nombre a mostrar del oro (mostramos Gold pero podria ponerse un nombre mas tuneado*/</i>	
114	const std::string &configGetGoldName() const ;	
115		
116	<i>/*Retorna la velocidad del player para moverse*/</i>	
117	unsigned int configPlayerSpeed() const ;	
118		
119	<i>/*Retorna el tiempo (en segundos) para que el player recupere vida/mana*/</i>	
120	double configPlayerRecoveryTime() const ;	
121		
122	const std::string& configPort() const ;	
123		

jul 21, 20 15:47	Configuration.h	Page 3/3
124	const std::string& configMapPath() const ;	
125		
126	const std::string& configSavePath() const ;	
127		
128	const std::string& configIndexPath() const ;	
129		
130	private:	
131	Configuration();	
132	};	
133		
134		
135	#endif //ARGENTUM_CONFIGURATION_H	

jul 21, 20 15:47

Configuration.cpp

Page 1/3

```

1  //
2  // Created by ivan on 8/6/20.
3  //
4
5  #include "Configuration.h"
6
7  using namespace GameType;
8
9  Configuration& Configuration::getInstance() {
10     static Configuration instance;
11     return instance;
12 }
13
14 Configuration::Configuration() {
15     Config::ConfigFileReader fileReader("/etc/Argentum/config.json");
16
17     fileReader.loadClassModifiers(classModifiers);
18     fileReader.loadRaceModifiers(raceModifiers);
19     fileReader.loadMonsterStats(monsterStats);
20
21     fileReader.loadWeaponData(weaponData);
22     fileReader.loadClothingData(clothingData);
23
24     fileReader.loadGoldModifiers(goldModifiers);
25     fileReader.loadXPModifiers(xpModifiers);
26
27     fileReader.loadPotionData(potionData);
28
29     fileReader.loadMonsterSpawnData(maxMonsterAmount, timeBetweenMonsterSpawns,
30                                     monsterSpawnAmount);
31
32     initialMerchantGold = fileReader.loadInitialMerchantGold();
33
34     criticalAttackChance = fileReader.loadCritAttackChance();
35     dodgeChance = fileReader.loadDodgeChance();
36
37     newbieLevel = fileReader.loadNewbieLevel();
38     maxLevelDif = fileReader.loadmaxLevelDif();
39     playerSpeed = fileReader.loadPlayerSpeed();
40     timeForPlayerRecovery = fileReader.loadTimeForPlayerRecovery();
41     port = fileReader.loadPort();
42     mapPath = fileReader.loadMapPath();
43     savePath = fileReader.loadSavePath();
44     indexPath = fileReader.loadIndexPath();
45     goldName = "Gold";
46 }
47
48 const Config::Modifiers& Configuration::configClassModifiers(Class _class) const
49 {
50     return classModifiers.at(_class);
51 }
52
53 const Config::Modifiers& Configuration::configRaceModifiers(Race race) const{
54     return raceModifiers.at(race);
55 }
56
57 const Config::MonsterStats& Configuration::configMonsterStats(GameType::Entity m
58 onster) const{
59     return monsterStats.at(monster);
60 }
61
62 const Config::WeaponData& Configuration::configWeaponData(Weapon weapon) const{
63     return weaponData.at(weapon);
64 }
65
66 const Config::ClothingData& Configuration::configClothingData(Clothing clothes)

```

jul 21, 20 15:47

Configuration.cpp

Page 2/3

```

65     const{
66         return clothingData.at(clothes);
67     }
68
69     const Config::GoldModifiers& Configuration::configGoldModifiers() const{
70         return goldModifiers;
71     }
72
73     const Config::XPModifiers& Configuration::configXPModifiers() const{
74         return xpModifiers;
75     }
76
77     const Config::PotionData& Configuration::configPotionData(Potion potion) const{
78         return potionData.at(potion);
79     }
80
81     float Configuration::configCriticalAttackChance() const{
82         return criticalAttackChance;
83     }
84
85     float Configuration::configDodgeChance() const{
86         return dodgeChance;
87     }
88
89     unsigned int Configuration::configNewbieLevel() const{
90         return newbieLevel;
91     }
92
93     unsigned int Configuration::configMaxLevelDif() const{
94         return maxLevelDif;
95     }
96
97     unsigned int Configuration::configTimeBetweenMonsterSpawns() const {
98         return timeBetweenMonsterSpawns;
99     }
100
101     unsigned int Configuration::configMaxMonsterAmount() const {
102         return maxMonsterAmount;
103     }
104
105     unsigned int Configuration::configMonsterSpawnAmount() const {
106         return monsterSpawnAmount;
107     }
108
109     unsigned int Configuration::configInitialMerchantGold() const {
110         return initialMerchantGold;
111     }
112
113     const std::string& Configuration::configGetGoldName() const {
114         return goldName;
115     }
116
117     const std::unordered_map<GameType::Weapon, Config::WeaponData> &
118         Configuration::configAllWeaponsData() {
119         return weaponData;
120     }
121
122     const std::unordered_map<GameType::Clothing, Config::ClothingData> &
123         Configuration::configAllClothingData() {
124         return clothingData;
125     }
126
127     const std::unordered_map<GameType::Potion, Config::PotionData> &
128         Configuration::configAllPotionsData() {
129         return potionData;
130     }

```

jul 21, 20 15:47

Configuration.cpp

Page 3/3

```

130
131 unsigned int Configuration::configPlayerSpeed() const {
132     return playerSpeed;
133 }
134
135 double Configuration::configPlayerRecoveryTime() const {
136     return timeForPlayerRecovery;
137 }
138
139 const std::string &Configuration::configPort() const {
140     return port;
141 }
142
143 const std::string &Configuration::configMapPath() const {
144     return mapPath;
145 }
146
147 const std::string &Configuration::configSavePath() const {
148     return savePath;
149 }
150
151 const std::string &Configuration::configIndexPath() const {
152     return indexPath;
153 }

```

jul 21, 20 15:47

ConfigFileReader.h

Page 1/3

```

1 //
2 // Created by ivan on 8/6/20.
3 //
4
5 #ifndef ARGENTUM_CONFIGFILEREADER_H
6 #define ARGENTUM_CONFIGFILEREADER_H
7
8 #include <fstream>
9 #include <unordered_map>
10 #include "../libs/GameEnums.h"
11 #include "json.hpp"
12
13 namespace Config {
14
15     struct Modifiers {
16         unsigned int lifeMultiplier;
17         unsigned int manaMultiplier;
18         unsigned int constitution;
19         unsigned int intelligence;
20         unsigned int agility;
21         unsigned int strength;
22         unsigned int meditationRate;
23         unsigned int recoveryRate;
24     };
25
26     struct WeaponData {
27         std::string name;
28         int minDmg;
29         int maxDmg;
30         unsigned int manaConsumption;
31         unsigned int range;
32         unsigned int price;
33     };
34
35     struct ClothingData {
36         std::string name;
37         unsigned int minDefense;
38         unsigned int maxDefense;
39         unsigned int price;
40     };
41
42     struct MonsterStats {
43         int life;
44         unsigned int damage;
45         unsigned int constitution;
46         unsigned int agility;
47         unsigned int strength;
48         unsigned int rangeOfVision;
49         unsigned int minLevel;
50         unsigned int maxLevel;
51         unsigned int reactionSpeed;
52         unsigned int speed;
53     };
54
55     struct PotionData {
56         std::string name;
57         unsigned int recoveryValue;
58         unsigned int price;
59     };
60
61     struct GoldModifiers {
62         float goldDropFactorMin;
63         float goldDropFactorMax;
64         unsigned int safeGoldFactor;
65         float safeGoldLevelModifier;
66     };

```

jul 21, 20 15:47

ConfigFileReader.h

Page 2/3

```

67     struct XPModifiers {
68         unsigned int nextLevelFactor;
69         float nextLevelModifier;
70         unsigned int attackXPModifier;
71         unsigned int killXPModifier;
72         float killXPMinRange;
73         float killXPMaxRange;
74     };
75
76
77     /*La clase ConfigFileReader es la encargada de parsear el archivo de configuracion de json que contiene los datos
78     * tuneables del juego*/
79
80     class ConfigFileReader {
81     private:
82         nlohmann::json obj;
83         std::unordered_map<std::string, GameType::Class> classes;
84         std::unordered_map<std::string, GameType::Race> races;
85         std::unordered_map<std::string, GameType::Entity> monsters;
86         std::unordered_map<std::string, GameType::Weapon> weapons;
87         std::unordered_map<std::string, GameType::Clothing> clothing;
88         std::unordered_map<std::string, GameType::Potion> potions;
89
90     public:
91         explicit ConfigFileReader(const std::string &path);
92
93         /*Carga los modificadores de las clases*/
94         void loadClassModifiers(std::unordered_map<GameType::Class, Modifiers> &mods);
95
96         /*Carga los modificadores de las razas*/
97         void loadRaceModifiers(std::unordered_map<GameType::Race, Modifiers> &mods);
98
99         /*Carga los stats de los monsters*/
100        void loadMonsterStats(std::unordered_map<GameType::Entity, MonsterStats> &stats);
101
102        /*Carga los datos de las armas*/
103        void loadWeaponData(std::unordered_map<GameType::Weapon, WeaponData> &stats);
104
105        /*Carga los datos de la ropa*/
106        void loadClothingData(std::unordered_map<GameType::Clothing, ClothingData> &stats);
107
108        /*Carga los modificadores del oro*/
109        void loadGoldModifiers(GoldModifiers &goldModifiers);
110
111        /*Carga los modificadores de la XP*/
112        void loadXPModifiers(XPModifiers &xpModifiers);
113
114        /*Carga la probabilidad de ataque critico*/
115        float loadCritAttackChance();
116
117        /*Carga la probabilidad de esquivar un ataque*/
118        float loadDodgeChance();
119
120        /*Carga el nivel de newbie de player*/
121        unsigned int loadNewbieLevel();
122
123        /*Carga la maxima diferencia de nivel permitida para el ataque entre players*/
124        unsigned int loadmaxLevelDif();
125

```

jul 21, 20 15:47

ConfigFileReader.h

Page 3/3

```

126        /*Carga la velocidad de los players para caminar*/
127        unsigned int loadPlayerSpeed();
128
129        /*Carga la data de las pociones (vida/mana recuperado)*/
130        void loadPotionData(std::unordered_map<GameType::Potion, PotionData> &stats);
131
132        /*Carga la data de spawn de los monsters (cada cuanto respawnean, cantidad que respawnean, etc)*/
133        void loadMonsterSpawnData(unsigned int &maxMonsterAmount,
134                                   unsigned int &timeBetweenMonsterSpawns,
135                                   unsigned int &monsterSpawnAmount);
136
137        /*Carga el oro inicial que tienen los mercaderes (para comprar items de los players)*/
138        unsigned int loadInitialMerchantGold();
139
140        /*Carga el tiempo (en segundos) que debe pasar para que el player recupere vida/mana*/
141        double loadTimeForPlayerRecovery();
142
143        std::string loadPort();
144        std::string loadMapPath();
145        std::string loadSavePath();
146        std::string loadIndexPath();
147
148    private:
149        static void
150        _getModifiers(Modifiers &modifier, nlohmann::json &currModifier);
151
152        static void
153        _getMonsterStats(MonsterStats &stats, nlohmann::json &currMonster);
154
155        static void
156        _getWeaponData(WeaponData &stats, nlohmann::json &currWeapon);
157
158        static void
159        _getClothingData(ClothingData &stats, nlohmann::json &currClothing);
160
161        static void _getPotionData(PotionData &stats, nlohmann::json &currPotion);
162    };
163 }
164
165 #endif //ARGENTUM_CONFIGFILEREADER_H

```


jul 21, 20 15:47

ConfigFileReader.cpp

Page 1/4

```

1  //
2  // Created by ivan on 8/6/20.
3  //
4
5  #include "ConfigFileReader.h"
6  #include "../libs/TPEException.h"
7  #include <memory>
8
9  using namespace GameType;
10 using json = nlohmann::json;
11
12 Config::ConfigFileReader::ConfigFileReader(const std::string& path) :
13     classes{{"Warrior", WARRIOR}, {"Wizard", WIZARD}, {"Paladin", PALADIN},
14             {"Cleric", CLERIC}},
15     races{{"Human", HUMAN}, {"Elf", ELF}, {"Dwarf", DWARF},
16           {"Gnome", GNOME}},
17     monsters{{"Skeleton", SKELETON}, {"Zombie", ZOMBIE}, {"Spider", SPIDER},
18             {"Goblin", GOBLIN}},
19     weapons{{"Longsword", LONGSWORD}, {"Axe", AXE}, {"Warhammer", WARHAMMER},
20            {"AshRod", ASH_ROD}, {"ElvenFlute", ELVEN_FLUTE},
21            {"LinkedStaff", LINKED_STAFF},
22            {"SimpleBow", SIMPLE_BOW}, {"CompositeBow", COMPOSITE_BOW},
23            {"GnarledStaff", GNARLED_STAFF},
24            {"Fist", FIST}},
25     clothing{{"CommonClothing", COMMON_CLOTHING}, {"LeatherArmor", LEATHER_ARMOR},
26             {"PlateArmor", PLATE_ARMOR},
27             {"KingArmor", KING_ARMOR}, {"BlueTunic", BLUE_TUNIC},
28             {"Hood", HOOD}, {"IronHelmet", IRON_HELMET},
29             {"TurtleShield", TURTLE_SHIELD}, {"IronShield", IRON_SHIELD},
30             {"MagicHat", MAGIC_HAT},
31             {"NoHelmet", NO_HELMET}, {"NoShield", NO_SHIELD}},
32     potions {{"HealthPotion", HEALTH_POTION}, {"ManaPotion", MANA_POTION}} {
33
34     std::ifstream file(path);
35     if (!file.is_open()) {
36         throw TPEException("Could not open Config File, check whether"
37                             " it exists or not");
38     }
39     try {
40         file >> obj;
41     } catch (...) {
42         throw TPEException("Json Config File parsing failed!");
43     }
44 }
45
46 void Config::ConfigFileReader::loadClassModifiers(std::unordered_map<Class, Modifiers>& mods) {
47     json& classModifiers = obj["Class"];
48     Modifiers currMods{};
49     for (auto & classModifier : classModifiers) {
50         _getModifiers(currMods, classModifier);
51         currMods.meditationRate = classModifier["MeditationRate"].get<uint>();
52         currMods.recoveryRate = 0;
53         mods.emplace(classes.at(classModifier["Name"]).get<std::string>(), currMods);
54     }
55 }
56
57 void Config::ConfigFileReader::loadRaceModifiers(std::unordered_map<Race, Modifiers>& mods) {
58     json& raceModifiers = obj["Race"];
59     Modifiers currMods{};
60     for (auto & raceModifier : raceModifiers) {
61         _getModifiers(currMods, raceModifier);
62         currMods.recoveryRate = raceModifier["RecoveryRate"].get<uint>();
63         currMods.meditationRate = 0;

```

jul 21, 20 15:47

ConfigFileReader.cpp

Page 2/4

```

64         mods.emplace(races.at(raceModifier["Name"]).get<std::string>(), currMods);
65     }
66 }
67
68 void Config::ConfigFileReader::loadWeaponData(std::unordered_map<Weapon, WeaponData>& stats) {
69     json& weaponsStats = obj["Weapon"];
70     WeaponData currStats{};
71     for (auto & weaponStat : weaponsStats) {
72         _getWeaponData(currStats, weaponStat);
73         stats.emplace(weapons.at(weaponStat["Name"]).get<std::string>(), currStats);
74     }
75 }
76
77 void Config::ConfigFileReader::loadClothingData(std::unordered_map<Clothing, ClothingData>& stats) {
78     json& clothingsStats = obj["Clothing"];
79     ClothingData currStats{};
80     for (auto & clothingStat : clothingsStats) {
81         _getClothingData(currStats, clothingStat);
82         stats.emplace(clothing.at(clothingStat["Name"]).get<std::string>(), currStats);
83     }
84 }
85
86 void Config::ConfigFileReader::loadPotionData(std::unordered_map<Potion, PotionData>& stats) {
87     json& potionData = obj["Potion"];
88     PotionData currPotion{};
89     for (auto & potion : potionData) {
90         _getPotionData(currPotion, potion);
91         stats.emplace(potions.at(potion["Name"]).get<std::string>(), currPotion);
92     }
93 }
94
95 void Config::ConfigFileReader::loadMonsterStats(std::unordered_map<GameType::Entity, MonsterStats>& stats) {
96     json& monsterStats = obj["Monster"];
97     MonsterStats currStats{};
98     for (auto & monsterStat : monsterStats) {
99         _getMonsterStats(currStats, monsterStat);
100         stats.emplace(monsters.at(monsterStat["Name"]).get<std::string>(), currStats);
101     }
102 }
103
104 void Config::ConfigFileReader::loadGoldModifiers(GoldModifiers& goldModifiers) {
105     json& modifiers = obj["GoldModifiers"];
106     goldModifiers.safeGoldFactor = modifiers["MaxSafeGoldFactor"].get<uint>();
107     goldModifiers.safeGoldLevelModifier = modifiers["MaxGoldLevelModifier"].get<float>();
108     goldModifiers.goldDropFactorMin = modifiers["MinRange"].get<float>();
109     goldModifiers.goldDropFactorMax = modifiers["MaxRange"].get<float>();
110 }
111
112 void Config::ConfigFileReader::loadXPModifiers(XPModifiers& xpModifiers) {
113     json& modifiers = obj["XPModifiers"];
114     xpModifiers.attackXPModifier = modifiers["AttackXPModifier"].get<unsigned int>();
115     xpModifiers.killXPMinRange = modifiers["MinKillXPModifier"].get<float>();
116     xpModifiers.killXPMaxRange = modifiers["MaxKillXPModifier"].get<float>();
117     xpModifiers.nextLevelModifier = modifiers["NextLevelModifier"].get<float>();
118 }
119

```

jul 21, 20 15:47

ConfigFileReader.cpp

Page 3/4

```

120     xpModifiers.nextLevelFactor = modifiers["NextLevelFactor"].get<unsigned int>();
121     xpModifiers.killXPModifier = modifiers["KillXPModifier"].get<unsigned int>();
122 }
123
124 float Config::ConfigFileReader::loadCritAttackChance() {
125     return obj["CritAttackProb"].get<float>();
126 }
127
128 float Config::ConfigFileReader::loadDodgeChance() {
129     return obj["DodgeCoeff"].get<float>();
130 }
131
132 unsigned int Config::ConfigFileReader::loadNewbieLevel() {
133     return obj["NewbieLevel"].get<unsigned int>();
134 }
135
136 unsigned int Config::ConfigFileReader::loadmaxLevelDif() {
137     return obj["MaxLevelDif"].get<unsigned int>();
138 }
139
140 void Config::ConfigFileReader::_getModifiers(Modifiers& modifier, json& currModi
fier){
141     modifier.lifeMultiplier = currModifier["Life"].get<unsigned int>();
142     modifier manaMultiplier = currModifier["Mana"].get<unsigned int>();
143     modifier.constitution = currModifier["Constitution"].get<unsigned int>();
144     modifier.intelligence = currModifier["Intelligence"].get<unsigned int>();
145     modifier.agility = currModifier["Agility"].get<unsigned int>();
146     modifier.strength = currModifier["Strength"].get<unsigned int>();
147 }
148
149 void Config::ConfigFileReader::_getMonsterStats(MonsterStats& stats, json& currM
onster){
150     stats.life = currMonster["Life"].get<unsigned int>();
151     stats.damage = currMonster["Damage"].get<unsigned int>();
152     stats.rangeOfVision = currMonster["VisionRange"].get<unsigned int>();
153     stats.minLevel = currMonster["LevelMin"].get<unsigned int>();
154     stats.maxLevel = currMonster["LevelMax"].get<unsigned int>();
155     stats.constitution = currMonster["Constitution"].get<unsigned int>();
156     stats.agility = currMonster["Agility"].get<unsigned int>();
157     stats.strength = currMonster["Strength"].get<unsigned int>();
158     stats.reactionSpeed = currMonster["ReactionSpeed"].get<unsigned int>();
159     stats.speed = currMonster["Speed"].get<unsigned int>();
160 }
161
162 void Config::ConfigFileReader::_getWeaponData(WeaponData& stats, json& currWeapo
n){
163     stats.name = currWeapon["Name"].get<std::string>();
164     stats.maxDmg = currWeapon["MaxDmg"].get<int>();
165     stats.minDmg = currWeapon["MinDmg"].get<int>();
166     stats.manaConsumption = currWeapon["ManaConsumption"].get<unsigned int>();
167     stats.range = currWeapon["Range"].get<unsigned int>();
168     stats.price = currWeapon["Price"].get<unsigned int>();
169 }
170
171 void Config::ConfigFileReader::_getClothingData(ClothingData& stats, json&
currClothing){
172     stats.name = currClothing["Name"].get<std::string>();
173     stats.maxDefense = currClothing["MaxDefense"].get<unsigned int>();
174     stats.minDefense = currClothing["MinDefense"].get<unsigned int>();
175     stats.price = currClothing["Price"].get<unsigned int>();
176 }
177
178
179 void Config::ConfigFileReader::_getPotionData(PotionData& stats, json&
currPotion){
180     stats.name = currPotion["Name"].get<std::string>();
181     stats.recoveryValue = currPotion["RecoveryValue"].get<unsigned int>();
182

```

jul 21, 20 15:47

ConfigFileReader.cpp

Page 4/4

```

183     stats.price = currPotion["Price"].get<unsigned int>();
184 }
185
186 void Config::ConfigFileReader::loadMonsterSpawnData(unsigned int &maxMonsterAmou
nt,
187     unsigned int &timeBetweenMonstersS
188     unsigned int &monsterSpawnAmount)
189 {
190     json& data = obj["MonsterSpawnData"];
191     maxMonsterAmount = data["MaxAmount"].get<unsigned int>();
192     timeBetweenMonsterSpawns = data["TimeBetweenSpawns"].get<unsigned int>();
193     monsterSpawnAmount = data["SpawnAmount"].get<unsigned int>();
194 }
195
196 unsigned int Config::ConfigFileReader::loadInitialMerchantGold() {
197     return obj["InitialMerchantGold"].get<unsigned int>();
198 }
199
200 unsigned int Config::ConfigFileReader::loadPlayerSpeed() {
201     return obj["PlayerSpeed"].get<unsigned int>();
202 }
203
204 double Config::ConfigFileReader::loadTimeForPlayerRecovery() {
205     return obj["TimeForPlayerRecoveryInSeconds"].get<unsigned int>();
206 }
207
208 std::string Config::ConfigFileReader::loadPort() {
209     return obj["Port"].get<std::string>();
210 }
211
212 std::string Config::ConfigFileReader::loadMapPath() {
213     return obj["MapPath"].get<std::string>();
214 }
215
216 std::string Config::ConfigFileReader::loadSavePath() {
217     return obj["SavePath"].get<std::string>();
218 }
219
220 std::string Config::ConfigFileReader::loadIndexPath() {
221     return obj["IndexPath"].get<std::string>();
222 }

```

jul 21, 20 15:47	Calculator.h	Page 1/2
1	<i>//</i>	
2	<i>// Created by ivan on 10/6/20.</i>	
3	<i>//</i>	
4		
5	#ifndef ARGENTUM_CALCULATOR_H	
6	#define ARGENTUM_CALCULATOR_H	
7		
8		
9	class Calculator {	
10	public:	
11		
12	<i>/*Calcula la cantidad maxima de vida del player*/</i>	
13	static int calculateMaxLife(unsigned int constitution, unsigned int classLif	
14	eMultiplier,	
15	<i>unsigned int raceLifeMultiplier, unsigned int le</i>	
16	vel);	
17	<i>/*Calcula la cantidad maxima de mana del player*/</i>	
18	static unsigned int calculateMaxMana(unsigned int intelligence, unsigned int	
19	classManaMultiplier,	
20	<i>unsigned int raceManaMultiplier, unsigne</i>	
21	d int level);	
22	<i>/*Calcula el drop de oro dropeado por un monstruo al matarlo*/</i>	
23	static unsigned int calculateGoldDrop(unsigned int maxLife);	
24	<i>/*Calcula cantidad de oro en mano segura*/</i>	
25	static unsigned int calculateMaxSafeGold(unsigned int level);	
26	<i>/*Calcula el XP necesario para subir de nivel*/</i>	
27	static unsigned int calculateNextLevelXP(unsigned int level);	
28	<i>/*Calcula el XP ganado en el ataque*/</i>	
29	static unsigned int calculateAttackXP(int dmg, unsigned int	
30	myLevel, unsigned int ot	
31	herLevel);	
32	<i>/*Calcula la cantidad de XP ganada por el player al matar a otro NPC (player</i>	
33	<i>o monster*/</i>	
34	static unsigned int calculateKillXP(unsigned int myLevel, unsigned int other	
35	Level,	
36	<i>unsigned int othermaxLife);</i>	
37	<i>/*Calcula el danio inicial del player, este danio podra ser esquivado o dism</i>	
38	<i>inuido en base a la defensa del rival*/</i>	
39	static int calculateDamage(unsigned int strength, int weaponDamage);	
40	<i>/*Retorna si el npc o player pudo o no esquivar el ataque*/</i>	
41	static bool canDodge(unsigned int agility);	
42	<i>/*Retorna un int random entre minRage y maxRange*/</i>	
43	static int getRandomInt(int minRange, int maxRange);	
44	<i>/*Retorna true si el ataque fue critical, flase en caso contrario*/</i>	
45	static bool isCritical();	
46	<i>/*Retorna la cantidad de vida recupera en base al recovery rate del player y</i>	
47	<i>el tiempo pasado*/</i>	
48	static int lifeRecovered(unsigned int recoveryRate, double timeElapsed);	
49	<i>/*Retorna la cantidad de mana recuperada sin estar en estado de meditacion e</i>	
50	<i>n base al tiempo pasado*/</i>	
51	static unsigned int manaRecoveredNoMeditation(unsigned int recoveryRate, dou	
52	ble timeElapsed);	
53	<i>/*Retorna la cantidad de mana recuperado cuando se esta en estado de meditac</i>	
54		
55		

jul 21, 20 15:47	Calculator.h	Page 2/2
56	<i>ion en base al tiempo pasado*/</i>	
57	static unsigned int manaRecoveredWithMeditation(unsigned int meditationRate,	
58	<i>unsigned int intelligence, double tim</i>	
59	eElapsed);	
60	private:	
61	static float _getRandomFloat(float minRange, float maxRange);	
62	};	
63		
64	#endif //ARGENTUM_CALCULATOR_H	

jul 21, 20 15:47

Calculator.cpp

Page 1/2

```

1  //
2  // Created by ivan on 10/6/20.
3  //
4
5  #include "Calculator.h"
6  #include "Configuration.h"
7  #include <random>
8
9  using namespace Config;
10
11 bool Calculator::isCritical() {
12     Configuration& config = Configuration::getInstance();
13     std::random_device seed;
14     std::default_random_engine generator(seed());
15     std::bernoulli_distribution dist(config.configCriticalAttackChance());
16     return dist(generator);
17 }
18
19 int Calculator::calculateMaxLife(unsigned int constitution, unsigned int classLifeMultiplier,
20                                 unsigned int raceLifeMultiplier, unsigned int level) {
21     return static_cast<int>(constitution * classLifeMultiplier * raceLifeMultiplier * level);
22 }
23
24 unsigned int Calculator::calculateMaxMana(unsigned int intelligence, unsigned int classManaMultiplier,
25                                           unsigned int raceManaMultiplier, unsigned int level) {
26     return intelligence * classManaMultiplier * raceManaMultiplier * level;
27 }
28
29 unsigned int Calculator::calculateGoldDrop(unsigned int maxLife) {
30     float minRange = Configuration::getInstance().configGoldModifiers().goldDropFactorMin;
31     float maxRange = Configuration::getInstance().configGoldModifiers().goldDropFactorMax;
32     float randNum = _getRandomFloat(minRange, maxRange);
33     return static_cast<unsigned int>(randNum * static_cast<float>(maxLife));
34 }
35
36 unsigned int Calculator::calculateMaxSafeGold(unsigned int level) {
37     unsigned int multiplier = Configuration::getInstance().configGoldModifiers().safeGoldFactor;
38     float exponent = Configuration::getInstance().configGoldModifiers().safeGoldLevelModifier;
39     return (multiplier * static_cast<unsigned int>(pow(level, exponent)));
40 }
41
42 unsigned int Calculator::calculateNextLevelXP(unsigned int level) {
43     unsigned int multiplier = Configuration::getInstance().configXPModifiers().nextLevelFactor;
44     float exponent = Configuration::getInstance().configXPModifiers().nextLevelModifier;
45     return (multiplier * static_cast<unsigned int>(pow(level, exponent)));
46 }
47
48 unsigned int Calculator::calculateAttackXP(int dmg, unsigned int myLevel, unsigned int otherLevel) {
49     if (dmg < 0) {
50         return 0;
51     }
52 }

```

jul 21, 20 15:47

Calculator.cpp

Page 2/2

```

62     unsigned int modifier = Configuration::getInstance().configXPModifiers().attackXPModifier;
63
64     int multiplier = static_cast<int>(otherLevel - myLevel + modifier);
65     return (dmg * std::max(multiplier, 0));
66 }
67
68 unsigned int Calculator::calculateKillXP(unsigned int myLevel, unsigned int otherLevel, unsigned int othermaxLife) {
69     unsigned int modifier = Configuration::getInstance().configXPModifiers().killXPModifier;
70     int multiplier = static_cast<int>((otherLevel - myLevel + modifier));
71     float minRange = Configuration::getInstance().configXPModifiers().killXPMinRange;
72     float maxRange = Configuration::getInstance().configXPModifiers().killXPMaxRange;
73     float random = _getRandomFloat(minRange, maxRange);
74     return static_cast<unsigned int>((random * static_cast<float>(othermaxLife) * std::max(multiplier, 0)));
75 }
76
77 int Calculator::calculateDamage(unsigned int strength, int weaponDamage) {
78     return static_cast<int>(strength * weaponDamage);
79 }
80
81 bool Calculator::canDodge(unsigned int agility) {
82     float random = _getRandomFloat(0, 1);
83     Configuration& config = Configuration::getInstance();
84     return (pow(random, agility) < config.configDodgeChance());
85 }
86
87 float Calculator::_getRandomFloat(float minRange, float maxRange) {
88     std::random_device seed;
89     std::default_random_engine generator(seed());
90     std::uniform_real_distribution<float> dist(minRange, maxRange);
91     return dist(generator);
92 }
93
94 int Calculator::getRandomInt(int minRange, int maxRange) {
95     std::random_device seed;
96     std::default_random_engine generator(seed());
97     std::uniform_int_distribution<int> dist(minRange, maxRange);
98     return dist(generator);
99 }
100
101 int Calculator::lifeRecovered(unsigned int recoveryRate, double timeElapsed) {
102     return static_cast<int>(static_cast<double>(recoveryRate) * timeElapsed) * 10;
103 }
104
105 unsigned int Calculator::manaRecoveredNoMeditation(unsigned int recoveryRate, double timeElapsed) {
106     return static_cast<int>(static_cast<double>(recoveryRate) * timeElapsed);
107 }
108
109 unsigned int Calculator::manaRecoveredWithMeditation(unsigned int meditationRate, unsigned int intelligence, double timeElapsed) {
110     return static_cast<int>(static_cast<double>(meditationRate * intelligence) * timeElapsed);
111 }

```

jul 21, 20 15:47

TPEXception.h

Page 1/1

```

1  //
2  // Created by marcos on 6/6/20.
3  //
4
5  #ifndef ARGENTUM_TPEXCEPTION_H
6  #define ARGENTUM_TPEXCEPTION_H
7
8  #include <exception>
9  #define BUF_LEN 256
10
11 class TPEXception : public std::exception {
12 private:
13     char errorMessage[BUF_LEN]{};
14 public:
15     explicit TPEXception(const char *fmt, ...);
16     const char * what() const noexcept override;
17 };
18
19
20 #endif //ARGENTUM_TPEXCEPTION_H

```

jul 21, 20 15:47

TPEXception.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/6/20.
3  //
4
5  #include <cstdarg>
6  #include <cstdio>
7  #include "TPEXception.h"
8
9  TPEXception::TPEXception(const char *fmt, ...) {
10     va_list args;
11     va_start(args, fmt);
12     vsnprintf(errorMessage, BUF_LEN, fmt, args);
13     va_end(args);
14     errorMessage[BUF_LEN - 1] = 0;
15 }
16
17 const char *TPEXception::what() const noexcept {
18     return errorMessage;
19 }
20
21

```

jul 21, 20 15:47

Timer.h

Page 1/1

```

1  //
2  // Created by marcos on 13/7/20.
3  //
4
5  #ifndef ARGENTUM_TIMER_H
6  #define ARGENTUM_TIMER_H
7
8  #include <chrono>
9
10 class Timer {
11 private:
12     std::chrono::high_resolution_clock::time_point time1{};
13     std::chrono::high_resolution_clock::time_point time2{};
14
15 public:
16     //Starts the timer
17     void start();
18
19     //Returns time passed since start in milliseconds
20     double getTime();
21 };
22
23
24 #endif //ARGENTUM_TIMER_H

```

jul 21, 20 15:47

Timer.cpp

Page 1/1

```

1  //
2  // Created by marcos on 13/7/20.
3  //
4
5  #include "Timer.h"
6  using namespace std::chrono;
7
8  void Timer::start() {
9      time1 = high_resolution_clock::now();
10 }
11
12 double Timer::getTime() {
13     time2 = high_resolution_clock::now();
14     duration<double, std::milli> timeStep = time2 - time1;
15     return timeStep.count();
16 }

```

jul 21, 20 15:47

Thread.h

Page 1/1

```

1  #ifndef TP3TALLER_THREAD_H
2  #define TP3TALLER_THREAD_H
3
4  /*Esta clase es abstracta, las clases que heredan de ella pueden correr el
5   * metodo run en un thread nuevo*/
6
7  #include <thread>
8
9  class Thread {
10 private:
11     std::thread thread;
12
13 public:
14     /*Ejecuta en un nuevo thread al metodo run*/
15     void operator()();
16
17     /*Joinea el thread*/
18     virtual void join();
19
20     /*Libera el thread violentamente*/
21     void detach();
22
23     virtual ~Thread() = default;
24
25 protected:
26     /*Metodo abstracto, las clases hijas deben implementarlo*/
27     virtual void run() = 0;
28 };
29
30
31 #endif //TP3TALLER_THREAD_H

```

jul 21, 20 15:47

Thread.cpp

Page 1/1

```

1  #include "Thread.h"
2
3  void Thread::operator()() {
4      thread = std::thread(&Thread::run, this);
5  }
6
7  void Thread::join() {
8      thread.join();
9  }
10
11 void Thread::detach() {
12     thread.detach();
13 }

```

jul 21, 20 15:47

Socket.h

Page 1/1

```

1  #ifndef TP3TALLER_SOCKET_H
2  #define TP3TALLER_SOCKET_H
3
4  #include <cstdio>
5  #include <string>
6  #include "TPEException.h"
7
8  class Socket {
9  private:
10     int fd; /*File Descriptor*/
11
12  public:
13     Socket();
14     Socket(const Socket&) = delete; /*Borro los constructores por copia*/
15     Socket operator=(const Socket&) = delete;
16     Socket& operator=(Socket&) noexcept;
17     Socket(Socket& srcSocket) noexcept;
18
19     /*Conecta el cliente al servidor*/
20     void connect(std::string& host, std::string& port);
21
22     /*Acepta una nueva conexion, retornando el socket generado*/
23     Socket accept() const;
24
25     /*Bindea a un socket*/
26     void bind(const std::string& port);
27
28     /*Setea la cantidad maxima de clientes que se tendran en espera*/
29     void maxListen(int max) const;
30
31     /*Envia el mensaje*/
32     void send(const char* message, size_t length) const;
33
34     /*Recibe el mensaje*/
35     void receive(char* message, size_t length) const;
36
37     /*Cierra el socket*/
38     void close();
39
40     ~Socket();
41
42  private:
43     explicit Socket(int fd) : fd(fd) {}
44     static struct addrinfo* _getAddresses(std::string* host, const std::string*
port);
45 };
46
47
48 #endif //TP3TALLER_SOCKET_H

```

jul 21, 20 15:47

Socket.cpp

Page 1/2

```

1  #include "Socket.h"
2  #include <netdb.h>
3  #include <unistd.h>
4  #include "TPEException.h"
5  #include <cstring>
6
7  #define CONNECT_ERROR_MSG "Could not run. "
8  #define BIND_ERROR_MSG "Could not bind. "
9  #define ACCEPT_ERROR_MSG "Error in accept: "
10 #define SEND_ERROR_MSG "Error in send: "
11 #define RECV_ERROR_MSG "Error in recv: "
12 #define GETADDRINFO_ERROR_MSG "Error in getaddrinfo: %s"
13
14 struct addrinfo* Socket::_getAddresses(std::string* host, const std::string* por
t) {
15     struct addrinfo hints{}, *result;
16     int s; /*Para verificar errores*/
17     memset(&hints, 0, sizeof(struct addrinfo));
18     hints.ai_family = AF_INET;
19     hints.ai_socktype = SOCK_STREAM;
20     if (host) {
21         hints.ai_flags = 0; /*cliente*/
22         s = getaddrinfo(host->c_str(), port->c_str(), &hints, &result);
23     } else {
24         hints.ai_flags = AI_PASSIVE; /*server*/
25         s = getaddrinfo(nullptr, port->c_str(), &hints, &result);
26     }
27     if (s != 0) throw TPEException(GETADDRINFO_ERROR_MSG, gai_strerror(s));
28     return result;
29 }
30
31 void Socket::connect(std::string& host, std::string& port) {
32     struct addrinfo* addresses = _getAddresses(&host, &port);
33     struct addrinfo* rp;
34     for (rp = addresses; rp != nullptr; rp = rp->ai_next) {
35         fd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);
36         if (fd == -1)
37             continue;
38
39         if (::connect(fd, rp->ai_addr, rp->ai_addrlen) != -1)
40             break; /*Logre conectarme*/
41
42         ::close(fd);
43     }
44     freeaddrinfo(addresses);
45     if (rp == nullptr) {
46         throw TPEException(CONNECT_ERROR_MSG);
47     }
48 }
49
50 void Socket::bind(const std::string& port) {
51     struct addrinfo* addresses = _getAddresses(nullptr, &port);
52     struct addrinfo* rp;
53     int val = 1;
54     for (rp = addresses; rp != nullptr; rp = rp->ai_next) {
55         fd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);
56         if (fd == -1)
57             continue;
58
59         setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &val, sizeof(val));
60
61         if (::bind(fd, rp->ai_addr, rp->ai_addrlen) == 0)
62             break; /*Logre bindear*/
63
64         ::close(fd);
65     }

```


jul 21, 20 15:47

Socket.cpp

Page 2/2

```

66     freeaddrinfo(addresses);
67     if (rp == nullptr) {
68         throw TException(BIND_ERROR_MSG);
69     }
70 }
71
72 Socket Socket::accept() const {
73     int peerFd = ::accept(fd, nullptr, nullptr);
74     if (peerFd == -1) {
75         throw TException(ACCEPT_ERROR_MSG);
76     }
77     return Socket(peerFd);
78 }
79
80 void Socket::send(const char* message, size_t length) const {
81     size_t bytesSent = 0;
82     int s = 0;
83
84     while (bytesSent < length) {
85         s = ::send(fd, message + bytesSent, length - bytesSent, MSG_NOSIGNAL);
86         if (s < 1) throw TException(SEND_ERROR_MSG);
87         bytesSent += s;
88     }
89 }
90
91 void Socket::receive(char* message, size_t length) const {
92     size_t bytesReceived = 0;
93     int s = 0;
94
95     while (bytesReceived < length) {
96         s = recv(fd, message + bytesReceived, length - bytesReceived, 0);
97         if (s < 1) throw TException(RECV_ERROR_MSG);
98         bytesReceived += s;
99     }
100 }
101
102 void Socket::maxListen(int max) const {
103     listen(fd, max);
104 }
105
106 Socket::~Socket() {
107     close();
108 }
109
110 Socket::Socket(Socket& srcSocket) noexcept {
111     fd = srcSocket.fd;
112     srcSocket.fd = -1;
113 }
114
115 Socket& Socket::operator=(Socket& srcSocket) noexcept {
116     fd = srcSocket.fd;
117     srcSocket.fd = -1;
118     return *this;
119 }
120
121 void Socket::close() {
122     if (fd != -1) {
123         shutdown(fd, SHUT_RDWR);
124         ::close(fd);
125         fd = -1;
126     }
127 }
128
129 Socket::Socket() {
130     fd = -1;
131 }

```

jul 21, 20 15:47

SharedConstants.h

Page 1/1

```

1  //
2  // Created by marcos on 17/7/20.
3  //
4
5  #ifndef ARGENTUM_SHAREDCONSTANTS_H
6  #define ARGENTUM_SHAREDCONSTANTS_H
7
8  const unsigned int TILE_DISTANCE_IN_METERS = 2000;
9
10 #endif //ARGENTUM_SHAREDCONSTANTS_H

```

jul 21, 20 15:47

GameEnums.h

Page 1/2

```

1  //
2  // Created by marcos on 20/6/20.
3  //
4
5  #ifndef ARGENTUM_GAMEENUMS_H
6  #define ARGENTUM_GAMEENUMS_H
7
8  #include <cinttypes>
9
10 namespace GameType {
11
12     enum ConnectionResponse : int32_t {
13         ACCEPTED, INEXISTENT_PLAYER, UNAVAILABLE_PLAYER, UNKNOWN_SERVER_ERROR
14     };
15
16     enum PlayerEvent : int32_t {
17         PLAYER_START_MOVING, PLAYER_STOP_MOVING, CREATE_PLAYER, LOAD_PLAYER, PLA
18 YER_ATTACK, PLAYER_USE_ITEM, PLAYER_UNEQUIP,
19         PLAYER_PICK_UP, PLAYER_DROP, PLAYER_LIST, PLAYER_BUY, PLAYER_SELL, PLAYE
20 R_WITHDRAW,
21         PLAYER_DEPOSIT, PLAYER_MEDITATE, PLAYER_RESURRECT, PLAYER_HEAL, PLAYER_S
22 END_MSG,
23         PLAYER_REQUEST_INVENTORY_NAMES
24     };
25
26     enum EventID: int32_t {
27         MOVED, ATTACK, UNEQUIP, EQUIPPED, CREATE_ENTITY, CREATE_ITEM, REMOVE_ENT
28 ITY,
29         DESTROY_ITEM, PLAYER_DEATH, RESURRECTED, TELEPORTED, PLAYER_LEVEL_UP
30     };
31
32     enum Race : int32_t {
33         HUMAN, ELF, DWARF, GNOME
34     };
35
36     enum Class : int32_t {
37         WIZARD, CLERIC, PALADIN, WARRIOR
38     };
39
40     enum Entity: int32_t {
41         SKELETON, ZOMBIE, SPIDER, GOBLIN, BANKER, GUARD, TRADER, PRIEST, PLAYER,
42 NOTHING
43     };
44
45     enum ItemType: int32_t {
46         ITEM_TYPE_GOLD, ITEM_TYPE_WEAPON, ITEM_TYPE_CLOTHING, ITEM_TYPE_POTION,
47 ITEM_TYPE_NONE
48     };
49
50     enum Weapon: int32_t {
51         LONGSWORD, AXE, WARHAMMER, ASH_ROD, ELVEN_FLUTE, LINKED_STAFF,
52 SIMPLE_BOW, COMPOSITE_BOW, GNARLED_STAFF, FIST, ZOMBIE_ATTACK, SPIDER_AT
53 TACK,
54         GOBLIN_ATTACK, SKELETON_ATTACK
55     };
56
57     enum Clothing: int32_t {
58         COMMON_CLOTHING, LEATHER_ARMOR, PLATE_ARMOR, KING_ARMOR, BLUE_TUNIC, HOO
59 D,
60         IRON_HELMET, TURTLE_SHIELD, IRON_SHIELD, MAGIC_HAT, NO_HELMET,
61 NO_SHIELD
62     };
63
64     enum Potion: int32_t {
65         HEALTH_POTION, MANA_POTION
66     };
67
68 }

```

jul 21, 20 15:47

GameEnums.h

Page 2/2

```

59
60     enum FloorType: int32_t {
61         GRASS0, GRASS1, GRASS2, GRASS3, SAND, WATER0, WATER1, WATER2, WATER3,
62         PRETTY_ROAD0, PRETTY_ROAD1, PRETTY_ROAD2, PRETTY_ROAD3, PRETTY_GRASS0,
63         PRETTY_GRASS1, PRETTY_GRASS2, PRETTY_GRASS3, DEAD_GRASS0, DEAD_GRASS1,
64         DEAD_GRASS2, DEAD_GRASS3, DARK_WATER0, DARK_WATER1, DARK_WATER2, DARK_WA
65 TER3,
66     };
67
68     enum Structure: int32_t {
69         BONE_GUY, BROKEN_RIP_STONE, BUSH, DEAD_BUSH, DEAD_GUY, DEAD_TREE,
70         FAT_TREE, HANGED_GUY, HOUSE1, HOUSE2, HOUSE3, LONG_TREE, PALM_TREE,
71         RIP_STONE, TREE, VERY_DEAD_GUY, SUNKEN_COLUMN, SUNKEN_SHIP, NO_STRUCTURE
72     };
73
74     enum Direction : int32_t {
75         DIRECTION_UP, DIRECTION_DOWN, DIRECTION_LEFT, DIRECTION_RIGHT, DIRECTION
76 _STILL
77     };
78
79     enum EquipmentPlace: int32_t {
80         EQUIPMENT_PLACE_NONE, EQUIPMENT_PLACE_HEAD, EQUIPMENT_PLACE_CHEST, EQUIP
81 MENT_PLACE_WEAPON,
82         EQUIPMENT_PLACE_SHIELD
83     };
84
85 #endif //ARGENTUM_GAMEENUMS_H

```

jul 21, 20 15:47

UpdateTeleportEntity.h

Page 1/1

```
1  //
2  // Created by marcos on 7/9/20.
3  //
4
5  #ifndef ARGENTUM_UPDATETELEPORTENTITY_H
6  #define ARGENTUM_UPDATETELEPORTENTITY_H
7
8  #include "UpdateEvent.h"
9  #include <string>
10 #include "../Map/Coordinate.h"
11
12 class UpdateTeleportEntity : public UpdateEvent {
13 private:
14     std::string nickname;
15     Coordinate newPosition;
16
17 public:
18     UpdateTeleportEntity(std::string^ _nickname, Coordinate _position) :
19         nickname(_nickname), newPosition(_position) {}
20
21     void operator()(GameGUI& game) override;
22 };
23
24
25 #endif //ARGENTUM_UPDATETELEPORTENTITY_H
```

jul 21, 20 15:47

UpdateTeleportEntity.cpp

Page 1/1

```
1  //
2  // Created by marcos on 7/9/20.
3  //
4
5  #include "UpdateTeleportEntity.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdateTeleportEntity::operator()(GameGUI &game) {
9      bool isMyPlayer = (game.getPlayerInfo().getNickname() == nickname);
10     game.getMap().teleportEntity(nickname, newPosition, isMyPlayer);
11 }
```

jul 21, 20 15:47

UpdateRemoveEntity.h

Page 1/1

```
1 //
2 // Created by marcos on 7/3/20.
3 //
4
5 #ifndef ARGENTUM_UPDATEREMOVEENTITY_H
6 #define ARGENTUM_UPDATEREMOVEENTITY_H
7
8 #include "UpdateEvent.h"
9 #include <string>
10
11 class UpdateRemoveEntity : public UpdateEvent {
12 private:
13     std::string nickname;
14
15 public:
16     explicit UpdateRemoveEntity(std::string^ _nickname) : nickname(_nickname) {
17     }
18     void operator()(GameGUI& game) override;
19 };
20
21 #endif //ARGENTUM_UPDATEREMOVEENTITY_H
```

jul 21, 20 15:47

UpdateRemoveEntity.cpp

Page 1/1

```
1 //
2 // Created by marcos on 7/3/20.
3 //
4
5 #include "UpdateRemoveEntity.h"
6 #include "../Client/GameGUI.h"
7
8 void UpdateRemoveEntity::operator()(GameGUI &game) {
9     game.getMap().removeEntity(nickname);
10 }
```

jul 21, 20 15:47

UpdatePlayerResurrect.h

Page 1/1

```
1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEPLAYERRESURRECT_H
6  #define ARGENTUM_UPDATEPLAYERRESURRECT_H
7
8  #include "UpdateEvent.h"
9  #include <string>
10
11 class UpdatePlayerResurrect : public UpdateEvent {
12 private:
13     std::string nickname;
14
15 public:
16     explicit UpdatePlayerResurrect(std::string^ _nickname) : nickname(std::move
17     (_nickname)) {}
18     void operator()(GameGUI& game);
19 };
20
21 #endif //ARGENTUM_UPDATEPLAYERRESURRECT_H
```

jul 21, 20 15:47

UpdatePlayerResurrect.cpp

Page 1/1

```
1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #include "UpdatePlayerResurrect.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdatePlayerResurrect::operator()(GameGUI &game) {
9      game.getMap().revivePlayer(nickname);
10 }
```

jul 21, 20 15:47

UpdatePlayerDeath.h

Page 1/1

```
1 //
2 // Created by marcos on 7/5/20.
3 //
4
5 #ifndef ARGENTUM_UPDATEPLAYERDEATH_H
6 #define ARGENTUM_UPDATEPLAYERDEATH_H
7
8 #include "UpdateEvent.h"
9 #include <string>
10
11 class UpdatePlayerDeath : public UpdateEvent {
12 private:
13     std::string nickname;
14
15 public:
16     explicit UpdatePlayerDeath(std::string^ _nickname) : nickname(std::move(_ni
17 ckname)) {}
18     void operator()(GameGUI& game);
19 };
20
21 #endif //ARGENTUM_UPDATEPLAYERDEATH_H
```

jul 21, 20 15:47

UpdatePlayerDeath.cpp

Page 1/1

```
1 //
2 // Created by marcos on 7/5/20.
3 //
4
5 #include "UpdatePlayerDeath.h"
6 #include "../Client/GameGUI.h"
7
8 void UpdatePlayerDeath::operator()(GameGUI &game) {
9     game.getMap().killPlayer(nickname);
10 }
```

jul 21, 20 15:47

UpdateMove.h

Page 1/1

```

1  //
2  // Created by marcos on 6/29/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEMOVE_H
6  #define ARGENTUM_UPDATEMOVE_H
7
8  #include "UpdateEvent.h"
9  #include <string>
10 #include "../libs/GameEnums.h"
11
12 class UpdateMove : public UpdateEvent {
13 private:
14     std::string nickname;
15     GameType::Direction direction;
16     unsigned int distanceTravelled;
17     bool reachedDestination;
18
19 public:
20     UpdateMove(std::string^ _nickname, GameType::Direction _direction,
21               unsigned int _distanceTravelled, bool _reachedDestination) :
22         nickname(std::move(_nickname)), direction(_direction), distanceTravelled(_distanceTravelled),
23         reachedDestination(_reachedDestination){}
24
25     void operator()(GameGUI& game) override;
26 };
27
28
29 #endif //ARGENTUM_UPDATEMOVE_H

```

jul 21, 20 15:47

UpdateMove.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/29/20.
3  //
4
5  #include "UpdateMove.h"
6  #include "../Map/Map.h"
7
8  #include "../Client/GameGUI.h"
9
10 void UpdateMove::operator()(GameGUI& game) {
11     game.getMap().moveEntity(nickname, direction, distanceTravelled,
12                             reachedDestination);
13 }

```

jul 21, 20 15:47

UpdateLevelUp.h

Page 1/1

```

1  //
2  // Created by marcos on 20/7/20.
3  //
4
5  #ifndef ARGENTUM_UPDATELEVELUP_H
6  #define ARGENTUM_UPDATELEVELUP_H
7
8  #include "UpdateEvent.h"
9  #include <string>
10
11 class UpdateLevelUp : public UpdateEvent {
12 private:
13     std::string playerNickname;
14     int level;
15
16 public:
17     explicit UpdateLevelUp(std::string^ _playerNickname, int _level) :
18         playerNickname(std::move(_playerNickname)), level(_level) {}
19     void operator()(GameGUI& game) override;
20 };
21
22
23 #endif //ARGENTUM_UPDATELEVELUP_H

```

jul 21, 20 15:47

UpdateLevelUp.cpp

Page 1/1

```

1  //
2  // Created by marcos on 20/7/20.
3  //
4
5  #include "UpdateLevelUp.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdateLevelUp::operator()(GameGUI &game) {
9      game.getMap().updatePlayerLevel(playerNickname, level);
10 }

```


jul 21, 20 15:47

UpdateGUI.h

Page 1/1

```

1  //
2  // Created by marcos on 7/2/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEGUI_H
6  #define ARGENTUM_UPDATEGUI_H
7
8  #include "UpdateEvent.h"
9  #include "../Client/EntityData.h"
10
11 class UpdateGUI : public UpdateEvent {
12 private:
13     PlayerData data;
14
15 public:
16     explicit UpdateGUI(PlayerData^ _data) : data(std::move(_data)) {}
17     void operator()(GameGUI& game) override;
18 };
19
20
21 #endif //ARGENTUM_UPDATEGUI_H

```

jul 21, 20 15:47

UpdateGUI.cpp

Page 1/1

```

1  //
2  // Created by marcos on 7/2/20.
3  //
4
5  #include "UpdateGUI.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdateGUI::operator()(GameGUI &game) {
9      for (const auto & item : data.equippedItems) {
10         game.getPlayerInventory().addEquipableItem(std::get<0>(item),
11                                                     std::get<1>(item));
12     }
13     for (const auto & item : data.inventoryItems) {
14         game.getPlayerInventory().addInventoryItem(std::get<0>(item),
15                                                    std::get<1>(item));
16     }
17     game.getPlayerInfo().update(data.generalInfo);
18     game.getMinichat().receiveText(data.minichatText);
19 }

```

jul 21, 20 15:47

UpdateEvent.h

Page 1/1

```

1  //
2  // Created by marcos on 6/29/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEEVENT_H
6  #define ARGENTUM_UPDATEEVENT_H
7
8  /*Interfaz, los eventos que updatean el juego deben implmenetar cada caso*/
9
10 class GameGUI;
11
12 class UpdateEvent {
13 public:
14     virtual void operator()(GameGUI& game) = 0;
15     virtual ~UpdateEvent() = default;
16 };
17
18
19 #endif //ARGENTUM_UPDATEEVENT_H

```

jul 21, 20 15:47

UpdateEquip.h

Page 1/1

```

1  //
2  // Created by marcos on 7/5/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEEQUIP_H
6  #define ARGENTUM_UPDATEEQUIP_H
7
8  #include "UpdateEvent.h"
9  #include <string>
10 #include "../libs/GameEnums.h"
11 #include "../Texture/TextureID.h"
12
13 const int UNEQUIP = -1; /*Uso esta constante para decirle que es un unequip*/
14
15 class UpdateEquip : public UpdateEvent {
16 private:
17     std::string nickname;
18     GameType::EquipmentPlace place;
19     TextureID equipment;
20
21 public:
22     UpdateEquip(std::string^ _nickname, GameType::EquipmentPlace _place,
23                 int32_t _equipment);
24
25     void operator()(GameGUI& game) override;
26 };
27
28
29 #endif //ARGENTUM_UPDATEEQUIP_H

```

jul 21, 20 15:47

UpdateEquip.cpp

Page 1/1

```

1  //
2  // Created by marcos on 7/5/20.
3  //
4
5  #include "UpdateEquip.h"
6  #include "../Client/ProtocolEnumTranslator.h"
7  #include "../Client/GameGUI.h"
8
9  UpdateEquip::UpdateEquip(std::string ^_nickname,
10                          GameType::EquipmentPlace _place, int32_t _equipment) {
11
12      ProtocolEnumTranslator translator;
13      nickname = std::move(_nickname);
14      place = _place;
15      if (_equipment == UNEQUIP) {
16          equipment = Nothing;
17      } else {
18          switch (place) {
19              case GameType::EQUIPMENT_PLACE_WEAPON:
20                  equipment = translator.getWeaponTexture(static_cast<GameType::We
21                  apon>(_equipment));
22                  break;
23
24              default:
25                  equipment = translator.getClothingTexture(static_cast<GameType::
26                  Clothing>(_equipment));
27          }
28      }
29
30      void UpdateEquip::operator()(GameGUI &game) {
31          game.getMap().equipOnPlayer(nickname, place, equipment);
32      }

```

jul 21, 20 15:47

UpdateDestroyItem.h

Page 1/1

```

1  //
2  // Created by marcos on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEDESTROYITEM_H
6  #define ARGENTUM_UPDATEDESTROYITEM_H
7
8  #include "UpdateEvent.h"
9  #include "../Map/Coordinate.h"
10
11 class UpdateDestroyItem : public UpdateEvent {
12 private:
13     Coordinate position;
14
15 public:
16     explicit UpdateDestroyItem(Coordinate _position) : position(_position) {}
17     void operator()(GameGUI& game) override;
18 };
19
20
21 #endif //ARGENTUM_UPDATEDESTROYITEM_H

```

jul 21, 20 15:47

UpdateDestroyItem.cpp

Page 1/1

```

1  //
2  // Created by marcos on 9/7/20.
3  //
4
5  #include "UpdateDestroyItem.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdateDestroyItem::operator()(GameGUI &game) {
9      game.getMap().destroyItem(position);
10 }

```

jul 21, 20 15:47

UpdateCreatePlayer.h

Page 1/1

```

1  //
2  // Created by ivan on 1/7/20.
3  //
4
5  #ifndef ARGENTUM_UPDATECREATEPLAYER_H
6  #define ARGENTUM_UPDATECREATEPLAYER_H
7
8  #include "UpdateEvent.h"
9  #include <string>
10 #include "../libs/GameEnums.h"
11 #include "../Client/ClientProtocol.h"
12
13 class UpdateCreatePlayer : public UpdateEvent{
14 private:
15     MapPlayerData data;
16 public:
17     explicit UpdateCreatePlayer(MapPlayerData& _data) : data(_data) {}
18     void operator()(GameGUI& game) override;
19 };
20
21
22 #endif //ARGENTUM_UPDATECREATEPLAYER_H

```

jul 21, 20 15:47

UpdateCreatePlayer.cpp

Page 1/1

```
1  //
2  // Created by ivan on 1/7/20.
3  //
4
5  #include "UpdateCreatePlayer.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdateCreatePlayer::operator()(GameGUI& game) {
9      game.addPlayer(data);
10 }
```

jul 21, 20 15:47

UpdateCreateNPC.h

Page 1/1

```
1  //
2  // Created by marcos on 7/2/20.
3  //
4
5  #ifndef ARGENTUM_UPDATECREATENPC_H
6  #define ARGENTUM_UPDATECREATENPC_H
7
8  #include "UpdateEvent.h"
9  #include "../Client/ClientProtocol.h"
10
11 class UpdateCreateNPC : public UpdateEvent {
12 private:
13     EntityData data;
14 public:
15     explicit UpdateCreateNPC(EntityData& _data) : data(_data) {}
16     void operator()(GameGUI& game) override;
17 };
18
19 #endif //ARGENTUM_UPDATECREATENPC_H
```

jul 21, 20 15:47

UpdateCreateNPC.cpp

Page 1/1

```

1  //
2  // Created by marcos on 7/2/20.
3  //
4
5  #include "UpdateCreateNPC.h"
6  #include "../Client/GameGUI.h"
7
8  void UpdateCreateNPC::operator()(GameGUI& game) {
9      game.addNPC(data);
10 }

```

jul 21, 20 15:47

UpdateCreateItem.h

Page 1/1

```

1  //
2  // Created by marcos on 7/5/20.
3  //
4
5  #ifndef ARGENTUM_UPDATECREATEITEM_H
6  #define ARGENTUM_UPDATECREATEITEM_H
7
8  #include "UpdateEvent.h"
9  #include "../libs/GameEnums.h"
10 #include "../Map/Coordinate.h"
11 #include "../Texture/TextureID.h"
12
13 class UpdateCreateItem : public UpdateEvent {
14 private:
15     TextureID item;
16     Coordinate position{};
17
18 public:
19     UpdateCreateItem(GameType::ItemType _type, int32_t _item, Coordinate _position);
20     void operator()(GameGUI& game) override;
21 };
22
23
24 #endif //ARGENTUM_UPDATECREATEITEM_H

```

jul 21, 20 15:47

UpdateCreateItem.cpp

Page 1/1

```

1  //
2  // Created by marcos on 7/5/20.
3  //
4
5  #include "UpdateCreateItem.h"
6  #include "../Client/GameGUI.h"
7  #include "../Client/ProtocolEnumTranslator.h"
8
9  UpdateCreateItem::UpdateCreateItem(GameType::ItemType _type, int32_t _item,
10                                     Coordinate _position) {
11      ProtocolEnumTranslator translator;
12      position = _position;
13      try {
14          switch (_type) {
15              case GameType::ITEM_TYPE_GOLD:
16                  item = Gold;
17                  break;
18              case GameType::ITEM_TYPE_WEAPON:
19                  item = translator.getWeaponDropTexture(static_cast<GameType::Wea
pon>(_item));
20                  break;
21              case GameType::ITEM_TYPE_CLOTHING:
22                  item = translator.getClothingDropTexture(static_cast<GameType::C
lothing>(_item));
23                  break;
24              case GameType::ITEM_TYPE_POTION:
25                  item = translator.getPotionTexture(static_cast<GameType::Potion>
(_item));
26                  break;
27              case GameType::ITEM_TYPE_NONE:
28                  throw TPEException("Tried to create a null item");
29          }
30      } catch (std::exception& e) {
31          std::cerr << e.what() << "Tried to create an unknown item" << std::endl;
32      }
33  }
34
35  void UpdateCreateItem::operator()(GameGUI &game) {
36      game.getMap().createItem(position, item);
37  }

```

jul 21, 20 15:47

UpdateAttack.h

Page 1/1

```

1  //
2  // Created by marcos on 7/7/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEATTACK_H
6  #define ARGENTUM_UPDATEATTACK_H
7
8  #include "UpdateEvent.h"
9  #include "../Map/Coordinate.h"
10 #include "../libs/GameEnums.h"
11 #include <string>
12
13 class UpdateAttack : public UpdateEvent {
14 private:
15     Coordinate position{};
16     GameType::Weapon weapon;
17     GameType::Direction attackDir;
18     std::string nickname;
19
20 public:
21     UpdateAttack(std::string& _nickname, Coordinate _position, int32_t _weapon,
GameType::Direction _attackDir);
22     void operator()(GameGUI& game) override;
23
24 private:
25     bool _shouldPlaySound(); //Para no spammeear siempre sonidos de ataque de los
monstruos
26 };
27
28
29
30 #endif //ARGENTUM_UPDATEATTACK_H

```

jul 21, 20 15:47

UpdateAttack.cpp

Page 1/2

```

1  //
2  // Created by marcos on 7/7/20.
3  //
4
5  #include "UpdateAttack.h"
6  #include "../Client/GameGUI.h"
7  #include <random>
8
9  #define ATTACK_HEARING_DISTANCE 6
10
11 const float MONSTER_SOUND_PROBABILITY = 0.15;
12
13 void UpdateAttack::operator()(GameGUI &game) {
14     switch (weapon) {
15         case GameType::GNARLED_STAFF:
16             game.getMap().addSpell(position, MagicMissile);
17             game.getMap().verifyQueueSound(position, Explotion3Sound, ATTACK_HEA
RING_DISTANCE);
18             break;
19         case GameType::ELVEN_FLUTE:
20             game.getMap().addSpell(position, Heal);
21             game.getMap().verifyQueueSound(position, HealingSound, ATTACK_HEARIN
G_DISTANCE);
22             break;
23         case GameType::LINKED_STAFF:
24             game.getMap().addSpell(position, Explosion);
25             game.getMap().verifyQueueSound(position, Explotion1Sound, ATTACK_HEA
RING_DISTANCE);
26             break;
27         case GameType::ASH_ROD:
28             game.getMap().addSpell(position, MagicArrow);
29             game.getMap().verifyQueueSound(position, Explotion2Sound, ATTACK_HEA
RING_DISTANCE);
30             break;
31         case GameType::LONGSWORD:
32             game.getMap().verifyQueueSound(position, SwordAttackSound, ATTACK_HE
ARING_DISTANCE);
33             break;
34         case GameType::AXE:
35             game.getMap().verifyQueueSound(position, Attack2Sound, ATTACK_HEARIN
G_DISTANCE);
36             break;
37         case GameType::WARHAMMER:
38             game.getMap().verifyQueueSound(position, HeavyAttackSound, ATTACK_HE
ARING_DISTANCE);
39             break;
40         case GameType::COMPOSITE_BOW:
41             game.getMap().addArrow(nickname, position, CompositeArrow);
42             game.getMap().verifyQueueSound(position, ArrowSound, ATTACK_HEARING_
DISTANCE);
43             break;
44         case GameType::SIMPLE_BOW:
45             game.getMap().addArrow(nickname, position, SimpleArrow);
46             game.getMap().verifyQueueSound(position, ArrowSound, ATTACK_HEARING_
DISTANCE);
47             break;
48         case GameType::ZOMBIE_ATTACK:
49             if (!_shouldPlaySound())
50                 game.getMap().verifyQueueSound(position, ZombieSound, ATTACK_HEA
RING_DISTANCE);
51             break;
52         case GameType::SPIDER_ATTACK:
53             if (!_shouldPlaySound())
54                 game.getMap().verifyQueueSound(position, SpiderSound, ATTACK_HEA
RING_DISTANCE);
55             break;

```

jul 21, 20 15:47

UpdateAttack.cpp

Page 2/2

```

56     case GameType::SKELETON_ATTACK:
57         if (!_shouldPlaySound())
58             game.getMap().verifyQueueSound(position, SkeletonSound, ATTACK_H
EARING_DISTANCE);
59         break;
60     case GameType::GOBLIN_ATTACK:
61         if (!_shouldPlaySound())
62             game.getMap().verifyQueueSound(position, GoblinSound, ATTACK_HEA
RING_DISTANCE);
63         break;
64     default:
65         break;
66 }
67 game.getMap().changeEntityLookDirection(nickname, attackDir);
68
69 UpdateAttack::UpdateAttack(std::string& _nickname, Coordinate _position,
70                             int32_t _weapon, GameType::Direction _attackDir) {
71     nickname = std::move(_nickname);
72     position = _position;
73     weapon = static_cast<GameType::Weapon>(_weapon);
74     attackDir = _attackDir;
75 }
76
77 bool UpdateAttack::_shouldPlaySound() {
78     std::random_device seed;
79     std::default_random_engine generator(seed());
80     std::bernoulli_distribution dist(MONSTER_SOUND_PROBABILITY);
81     return dist(generator);
82 }
83

```


jul 21, 20 15:47

TextureRepository.h

Page 1/2

```

1  //
2  // Created by marcos on 9/6/20.
3  //
4
5  #ifndef ARGENTUM_TEXTUREREPOSITORY_H
6  #define ARGENTUM_TEXTUREREPOSITORY_H
7
8  #include <unordered_map>
9  #include "Texture.h"
10 #include "TextureID.h"
11
12 class TextureRepository {
13 private:
14     std::unordered_map<TextureID, Texture> textures;
15     SDL_Renderer& renderer;
16
17 public:
18     explicit TextureRepository(SDL_Renderer& renderer);
19     Texture& getTexture(TextureID texture);
20
21     /*Devuelve el mismo renderer que la window. No es ideal pero fue la mejor so
lucion
22     * para poder crear el texto del nivel/nombre de las entidades ya que necesi
tamos
23     * el renderer para text y sino tendríamos que pedirlo de screen, estando es
te
24     * mucho mas arriba en jerarquia y no es posible sin cambiar todo el modelo*
/
25     SDL_Renderer& getRenderer() const;
26
27 private:
28     void _loadClothing();
29     void _loadHeads();
30     void _loadWeapons();
31     void _loadTiles();
32     void _loadStructures();
33     void _loadNPCS();
34     void _loadDrops();
35     void _loadMiscellaneous();
36     void _loadUI();
37
38     void _setImage(TextureID TextureID, std::string^ image,
39                     int width, int height, int xOffset = 0, int yOffset = 0, int
scale = 1
40                     , ColorKey_t key = {0, 0, 0});
41     void _setSpellImage(TextureID TextureID, std::string^ spellImage,
42                         int width, int height, int xOffset = 0, int yOffset
= 0);
43     void _setNPCImage(TextureID TextureID, std::string^ npcImage, int width, in
t height
44                     , int xOffset = 0, int yOffset =
0);
45     void _setBodyImage(TextureID texture, std::string^ bodyImage);
46     void _setShieldImage(TextureID TextureID, std::string^ shieldImage);
47     void _setWeaponImage(TextureID TextureID, std::string^ weaponImage);
48     void _setTileImage(TextureID TextureID, std::string^ tileImage, bool indivi
dualTile);
49     void _setHeadImage(TextureID TextureID, std::string^ headImage);
50     void _setHelmetImage(TextureID TextureID, std::string^ helmetImage,
51                         int xOffset = 0, int yOffset = 0);
52
53     static void _addBodySprites(Texture& texture, int y, bool lateralSide);
54     static void _addWeaponSprites(Texture& texture, int y, bool lateralSide);
55     static void _addShieldSprites(Texture& texture, int y, bool lateralSide);
56     static void _addNPCSprites(Texture& texture, int y, bool lateralSide, int wi
dth, int height);

```

jul 21, 20 15:47

TextureRepository.h

Page 2/2

```

57     static void _addTileSprites(Texture& texture, int y, bool individualTile);
58     static void _addSprites(Texture& texture, int width, int height);
59     static void _addSpellSprites(Texture& texture, int y, int width, int height)
;
60 };
61
62
63 #endif //ARGENTUM_TEXTUREREPOSITORY_H

```

jul 21, 20 15:47

TextureRepository.cpp

Page 1/7

```

1  //
2  // Created by marcos on 9/6/20.
3  //
4
5  #include "TextureRepository.h"
6  #include "../Client/GameConstants.h"
7
8  #define PLAYER_GHOST_PATH "/var/Argentum/Assets/Images/Miscellaneous/PlayerGhost.png"
9  #define BLUE_TUNIC_PATH "/var/Argentum/Assets/Images/Clothing/BlueTunic.png"
10 #define BLUE_TUNIC_DROP_PATH "/var/Argentum/Assets/Images/Clothing/BlueTunicDrop.png"
11 #define COMMON_CLOTHING_PATH "/var/Argentum/Assets/Images/Clothing/CommonClothing.png"
12 #define COMMON_CLOTHING_DROP_PATH "/var/Argentum/Assets/Images/Clothing/CommonClothingDrop.png"
13 #define HOOD_PATH "/var/Argentum/Assets/Images/Clothing/Hood.png"
14 #define HOOD_DROP_PATH "/var/Argentum/Assets/Images/Clothing/HoodDrop.png"
15 #define IRON_HELMET_PATH "/var/Argentum/Assets/Images/Clothing/IronHelmet.png"
16 #define IRON_HELMET_DROP_PATH "/var/Argentum/Assets/Images/Clothing/IronHelmetDrop.png"
17 #define IRON_SHIELD_PATH "/var/Argentum/Assets/Images/Clothing/IronShield.png"
18 #define IRON_SHIELD_DROP_PATH "/var/Argentum/Assets/Images/Clothing/IronShieldDrop.png"
19 #define KING_ARMOR_PATH "/var/Argentum/Assets/Images/Clothing/KingArmor.png"
20 #define KING_ARMOR_DROP_PATH "/var/Argentum/Assets/Images/Clothing/KingArmorDrop.png"
21 #define LEATHER_ARMOR_PATH "/var/Argentum/Assets/Images/Clothing/LeatherArmor.png"
22 #define LEATHER_ARMOR_DROP_PATH "/var/Argentum/Assets/Images/Clothing/LeatherArmorDrop.png"
23 #define MAGIC_HAT_PATH "/var/Argentum/Assets/Images/Clothing/MagicHat.png"
24 #define MAGIC_HAT_DROP_PATH "/var/Argentum/Assets/Images/Clothing/MagicHatDrop.png"
25 #define PLATE_ARMOR_PATH "/var/Argentum/Assets/Images/Clothing/PlateArmor.png"
26 #define PLATE_ARMOR_DROP_PATH "/var/Argentum/Assets/Images/Clothing/PlateArmorDrop.png"
27 #define TURTLE_SHIELD_PATH "/var/Argentum/Assets/Images/Clothing/TurtleShield.png"
28 #define TURTLE_SHIELD_DROP_PATH "/var/Argentum/Assets/Images/Clothing/TurtleShieldDrop.png"
29 #define DWARF_HEAD_PATH "/var/Argentum/Assets/Images/Heads/DwarfHead.png"
30 #define ELF_HEAD_PATH "/var/Argentum/Assets/Images/Heads/ElfHead.png"
31 #define GNOME_HEAD_PATH "/var/Argentum/Assets/Images/Heads/GnomeHead.png"
32 #define HUMAN_HEAD_PATH "/var/Argentum/Assets/Images/Heads/HumanHead.png"
33 #define ASH_ROD_PATH "/var/Argentum/Assets/Images/Items/AshRod.png"
34 #define ASH_ROD_DROP_PATH "/var/Argentum/Assets/Images/Items/AshRodDrop.png"
35 #define AXE_PATH "/var/Argentum/Assets/Images/Items/Axe.png"
36 #define AXE_DROP_PATH "/var/Argentum/Assets/Images/Items/AxeDrop.png"
37 #define COMPOSITE_BOW_PATH "/var/Argentum/Assets/Images/Items/CompositeBow.png"
38 #define COMPOSITE_BOW_DROP_PATH "/var/Argentum/Assets/Images/Items/CompositeBowDrop.png"
39 #define ELVEN_FLUTE_DROP_PATH "/var/Argentum/Assets/Images/Items/ElvenFluteDrop.png"
40 #define LINKED_STAFF_PATH "/var/Argentum/Assets/Images/Items/LinkedStaff.png"
41 #define LINKED_STAFF_DROP_PATH "/var/Argentum/Assets/Images/Items/LinkedStaffDrop.png"
42 #define GNARLED_STAFF_PATH "/var/Argentum/Assets/Images/Items/GnarledStaff.png"
43 #define GNARLED_STAFF_DROP_PATH "/var/Argentum/Assets/Images/Items/GnarledStaffDrop.png"
44 #define LONG_SWORD_PATH "/var/Argentum/Assets/Images/Items/LongSword.png"
45 #define LONG_SWORD_DROP_PATH "/var/Argentum/Assets/Images/Items/LongSwordDrop.png"
46 #define SIMPLE_BOW_PATH "/var/Argentum/Assets/Images/Items/SimpleBow.png"
47 #define SIMPLE_BOW_DROP_PATH "/var/Argentum/Assets/Images/Items/SimpleBowDrop.png"
48 #define WAR_HAMMER_PATH "/var/Argentum/Assets/Images/Items/WarHammer.png"
49 #define WAR_HAMMER_DROP_PATH "/var/Argentum/Assets/Images/Items/WarHammerDrop.png"
50 #define HEALTH_POTION_PATH "/var/Argentum/Assets/Images/Items/HealthPotion.png"
51 #define MANA_POTION_PATH "/var/Argentum/Assets/Images/Items/ManaPotion.png"
52 #define GRASS_PATH "/var/Argentum/Assets/Images/Map/Grass.png"
53 #define PRETTY_GRASS_PATH "/var/Argentum/Assets/Images/Map/PrettyGrass.png"
54 #define PRETTY_ROAD_PATH "/var/Argentum/Assets/Images/Map/PrettyRoad.png"
55 #define DEAD_GRASS_PATH "/var/Argentum/Assets/Images/Map/DeadGrass.png"
56 #define SAND_PATH "/var/Argentum/Assets/Images/Map/Sand.png"
57 #define WATER_PATH "/var/Argentum/Assets/Images/Map/Water.png"
58 #define DARK_WATER_PATH "/var/Argentum/Assets/Images/Map/DarkWater.png"
59 #define SKELETON_PATH "/var/Argentum/Assets/Images/Monsters/Skeleton.png"
60 #define GOBLIN_PATH "/var/Argentum/Assets/Images/Monsters/Goblin.png"
61 #define ZOMBIE_PATH "/var/Argentum/Assets/Images/Monsters/Zombie.png"
62 #define SPIDER_PATH "/var/Argentum/Assets/Images/Monsters/Spider.png"
63 #define PRIEST_PATH "/var/Argentum/Assets/Images/Citizens/Priest.png"
64 #define TRADER_PATH "/var/Argentum/Assets/Images/Citizens/Trader.png"
65 #define BANKER_PATH "/var/Argentum/Assets/Images/Citizens/Banker.png"

```

jul 21, 20 15:47

TextureRepository.cpp

Page 2/7

```

66 #define GUARD_PATH "/var/Argentum/Assets/Images/Citizens/Guard.png"
67 #define TREE_PATH "/var/Argentum/Assets/Images/Map/Tree.png"
68 #define LONG_TREE_PATH "/var/Argentum/Assets/Images/Map/LongTree.png"
69 #define FAT_TREE_PATH "/var/Argentum/Assets/Images/Map/FatTree.png"
70 #define PALM_TREE_PATH "/var/Argentum/Assets/Images/Map/PalmTree.png"
71 #define DEAD_TREE_PATH "/var/Argentum/Assets/Images/Map/DeadTree.png"
72 #define BUSH_PATH "/var/Argentum/Assets/Images/Map/Bush.png"
73 #define DEAD_BUSH_PATH "/var/Argentum/Assets/Images/Map/DeadBush.png"
74 #define HOUSE1_PATH "/var/Argentum/Assets/Images/Map/House1.png"
75 #define HOUSE2_PATH "/var/Argentum/Assets/Images/Map/House2.png"
76 #define HOUSE3_PATH "/var/Argentum/Assets/Images/Map/House3.png"
77 #define SUNKEN_COLUMN_PATH "/var/Argentum/Assets/Images/Map/SunkenColumn.png"
78 #define SUNKEN_SHIP_PATH "/var/Argentum/Assets/Images/Map/SunkenShip.png"
79 #define BONE_GUY_PATH "/var/Argentum/Assets/Images/Map/BoneGuy.png"
80 #define BROKEN_RIP_STONE_PATH "/var/Argentum/Assets/Images/Map/BrokenRipStone.png"
81 #define DEAD_GUY_PATH "/var/Argentum/Assets/Images/Map/DeadGuy.png"
82 #define VERY_DEAD_GUY_PATH "/var/Argentum/Assets/Images/Map/VeryDeadGuy.png"
83 #define HANGED_GUY_PATH "/var/Argentum/Assets/Images/Map/HangedGuy.png"
84 #define RIP_STONE_PATH "/var/Argentum/Assets/Images/Map/RipStone.png"
85 #define EXPLOSION_PATH "/var/Argentum/Assets/Images/Spells/Explosion.png"
86 #define MAGIC_ARROW_PATH "/var/Argentum/Assets/Images/Spells/MagicArrow.png"
87 #define MAGIC_MISSILE_PATH "/var/Argentum/Assets/Images/Spells/MagicMissile.png"
88 #define HEAL_PATH "/var/Argentum/Assets/Images/Spells/Heal.png"
89 #define GOLD_PATH "/var/Argentum/Assets/Images/Miscellaneous/Gold.png"
90 #define SIMPLE_ARROW_PATH "/var/Argentum/Assets/Images/Miscellaneous/SimpleArrow.png"
91 #define COMPOSITE_ARROW_PATH "/var/Argentum/Assets/Images/Miscellaneous/CompositeArrow.png"
92 #define BACKGROUND_PATH "/var/Argentum/Assets/Images/UI/Background.png"
93 #define MAIN_MENU_PATH "/var/Argentum/Assets/Images/UI/MainMenuBackground.png"
94
95 TextureRepository::TextureRepository(SDL_Renderer& renderer) : renderer(renderer)
96 {
97     _loadClothing();
98     _loadHeads();
99     _loadWeapons();
100     _loadTiles();
101     _loadStructures();
102     _loadNPCs();
103     _loadDrops();
104     _loadMiscellaneous();
105     _loadUI();
106 }
107
108 void TextureRepository::_loadUI() {
109     _setImage(Background, BACKGROUND_PATH, 1495, 937, 0, 0, 1, {-1, -1, -1});
110     _setImage(MainMenu, MAIN_MENU_PATH, 1499, 937, 0, 0, 1, {-1, -1, -1});
111 }
112
113 void TextureRepository::_loadMiscellaneous() {
114     _setSpellImage(Explosion, EXPLOSION_PATH, 256, 256, -10, -10);
115     _setSpellImage(MagicArrow, MAGIC_ARROW_PATH, 96, 100, 20, 15);
116     _setSpellImage(MagicMissile, MAGIC_MISSILE_PATH, 128, 128, 8, 5);
117     _setSpellImage(Heal, HEAL_PATH, 76, 76, 25, 20);
118     _setImage(SimpleArrow, SIMPLE_ARROW_PATH, 32, 32, 45, 45, 1);
119     _setImage(CompositeArrow, COMPOSITE_ARROW_PATH, 32, 32, 45, 45, 1);
120 }
121
122 void TextureRepository::_loadDrops() {
123     _setImage(BlueTunicDrop, BLUE_TUNIC_DROP_PATH, 32, 32, 30, 30, 2);
124     _setImage(LongSwordDrop, LONG_SWORD_DROP_PATH, 32, 32, 33, 30, 2);
125     _setImage(LinkedStaffDrop, LINKED_STAFF_DROP_PATH, 32, 32, 30, 30, 2);
126     _setImage(GnarledStaffDrop, GNARLED_STAFF_DROP_PATH, 32, 32, 35, 30, 2);
127     _setImage(MagicHatDrop, MAGIC_HAT_DROP_PATH, 32, 32, 50, 45);
128     _setImage(HealthPotion, HEALTH_POTION_PATH, 32, 32, 50, 45);
129     _setImage(ManaPotion, MANA_POTION_PATH, 32, 32, 50, 45);
130 }

```

jul 21, 20 15:47

TextureRepository.cpp

Page 3/7

```

131 _setImage(CommonClothingDrop, COMMON_CLOTHING_DROP_PATH, 32, 32, 35, 30, 2);
132 _setImage(KingArmorDrop, KING_ARMOR_DROP_PATH, 32, 32, 35, 30, 2);
133 _setImage(LeatherArmorDrop, LEATHER_ARMOR_DROP_PATH, 32, 32, 35, 30, 2);
134 _setImage(PlateArmorDrop, PLATE_ARMOR_DROP_PATH, 16, 32, 48, 35, 2);
135 _setImage(HoodDrop, HOOD_DROP_PATH, 32, 32, 50, 45);
136 _setImage(IronHelmetDrop, IRON_HELMET_DROP_PATH, 32, 32, 50, 45);
137 _setImage(IronShieldDrop, IRON_SHIELD_DROP_PATH, 32, 32, 35, 30, 2);
138 _setImage(TurtleShieldDrop, TURTLE_SHIELD_DROP_PATH, 32, 32, 50, 45);
139 _setImage(AshRodDrop, ASH_ROD_DROP_PATH, 32, 32, 35, 30, 2);
140 _setImage(AxeDrop, AXE_DROP_PATH, 32, 32, 32, 30, 2);
141 _setImage(CompositeBowDrop, COMPOSITE_BOW_DROP_PATH, 32, 32, 32, 30, 2);
142 _setImage(ElvenFluteDrop, ELVEN_FLUTE_DROP_PATH, 32, 32, 32, 30, 2);
143 _setImage(SimpleBowDrop, SIMPLE_BOW_DROP_PATH, 32, 32, 32, 30, 2);
144 _setImage(WarHammerDrop, WAR_HAMMER_DROP_PATH, 32, 32, 32, 28, 2);
145 _setImage(Gold, GOLD_PATH, 32, 32, 45, 50, 1);
146 }
147
148 void TextureRepository::_loadClothing() {
149     _setBodyImage(BlueTunic, BLUE_TUNIC_PATH);
150     _setBodyImage(CommonClothing, COMMON_CLOTHING_PATH);
151     _setShieldImage(IronShield, IRON_SHIELD_PATH);
152     _setBodyImage(LeatherArmor, LEATHER_ARMOR_PATH);
153     _setBodyImage(PlateArmor, PLATE_ARMOR_PATH);
154     _setBodyImage(KingArmor, KING_ARMOR_PATH);
155     _setShieldImage(TurtleShield, TURTLE_SHIELD_PATH);
156     _setHelmetImage(Hood, HOOD_PATH);
157     _setHelmetImage(IronHelmet, IRON_HELMET_PATH);
158     _setHelmetImage(MagicHat, MAGIC_HAT_PATH, -1, -24);
159 }
160
161 void TextureRepository::_loadHeads() {
162     _setHeadImage(DwarfHead, DWARF_HEAD_PATH);
163     _setHeadImage(ElfHead, ELF_HEAD_PATH);
164     _setHeadImage(GnomeHead, GNOME_HEAD_PATH);
165     _setHeadImage(HumanHead, HUMAN_HEAD_PATH);
166 }
167
168 void TextureRepository::_loadWeapons() {
169     _setWeaponImage(AshRod, ASH_ROD_PATH);
170     _setWeaponImage(Axe, AXE_PATH);
171     _setWeaponImage(CompositeBow, COMPOSITE_BOW_PATH);
172     _setWeaponImage(LinkedStaff, LINKED_STAFF_PATH);
173     _setWeaponImage(GnarledStaff, GNARLED_STAFF_PATH);
174     _setWeaponImage(LongSword, LONG_SWORD_PATH);
175     _setWeaponImage(SimpleBow, SIMPLE_BOW_PATH);
176     _setWeaponImage(WarHammer, WAR_HAMMER_PATH);
177 }
178
179 void TextureRepository::_loadTiles() {
180     _setTileImage(Grass, GRASS_PATH, false);
181     _setTileImage(PrettyGrass, PRETTY_GRASS_PATH, false);
182     _setTileImage(PrettyRoad, PRETTY_ROAD_PATH, false);
183     _setTileImage(DeadGrass, DEAD_GRASS_PATH, false);
184     _setTileImage(Water, WATER_PATH, false);
185     _setTileImage(DarkWater, DARK_WATER_PATH, false);
186     _setTileImage(Sand, SAND_PATH, true);
187 }
188
189 void TextureRepository::_loadStructures() {
190     _setImage(Tree, TREE_PATH, 256, 256, -60, -180);
191     _setImage(LongTree, LONG_TREE_PATH, 256, 256, -60, -180);
192     _setImage(FatTree, FAT_TREE_PATH, 256, 256, -60, -180);
193     _setImage(PalmTree, PALM_TREE_PATH, 256, 256, -60, -180);
194     _setImage(DeadTree, DEAD_TREE_PATH, 256, 256, -60, -160);
195     _setImage(Bush, BUSH_PATH, 75, 65, 35, 35);
196     _setImage(BoneGuy, BONE_GUY_PATH, 75, 65, 30, 40);

```

jul 21, 20 15:47

TextureRepository.cpp

Page 4/7

```

197 _setImage(BrokenRipStone, BROKEN_RIP_STONE_PATH, 75, 65, 30, 20);
198 _setImage(DeadGuy, DEAD_GUY_PATH, 75, 65, 30, -60, 2);
199 _setImage(VeryDeadGuy, VERY_DEAD_GUY_PATH, 75, 65, 0, 10, 2);
200 _setImage(HangedGuy, HANGED_GUY_PATH, 75, 65, 5, -60, 2);
201 _setImage(RipStone, RIP_STONE_PATH, 75, 65, 30, 40);
202 _setImage(DeadBush, DEAD_BUSH_PATH, 75, 65, 30, 40);
203 _setImage(House1, HOUSE1_PATH, 196, 200, 40, -150);
204 _setImage(House2, HOUSE2_PATH, 181, 213, 40, -150);
205 _setImage(House3, HOUSE3_PATH, 200, 239, 30, -165);
206 _setImage(SunkenShip, SUNKEN_SHIP_PATH, 256, 256, -120, -10, 2);
207 _setImage(SunkenColumn, SUNKEN_COLUMN_PATH, 256, 256, 5, -185);
208 }
209
210 void TextureRepository::_loadNPCs() {
211     _setNPCImage(Skeleton, SKELETON_PATH, 25, 52, 35, 30);
212     _setNPCImage(Goblin, GOBLIN_PATH, 24, 31, 38, 48);
213     _setNPCImage(Zombie, ZOMBIE_PATH, 25, 45, 35, 30);
214     _setNPCImage(Spider, SPIDER_PATH, 34, 34, 30, 48);
215     _setNPCImage(Priest, PRIEST_PATH, 25, 45, 37, 33);
216     _setNPCImage(Trader, TRADER_PATH, 24, 48, 37, 33);
217     _setNPCImage(Banker, BANKER_PATH, 25, 45, 37, 33);
218     _setNPCImage(Guard, GUARD_PATH, 28, 52, 37, 33);
219     _setNPCImage(PlayerGhost, PLAYER_GHOST_PATH, 47, 71, 20, -10); /*tiene el mi
smo formato*/
220 }
221
222 void TextureRepository::_setImage(TextureID TextureID, std::string^ image,
223     int width, int height, int xOffset, int yOffset, int scale,
224     ColorKey_t key) {
225     try {
226         textures.emplace(TextureID, renderer);
227         Texture& texture = textures.at(TextureID);
228         texture.loadFromFile(image, key, xOffset, yOffset, scale);
229         _addSprites(texture, width, height);
230     } catch (TPEException& e) {
231         throw TPEException("Failed to load %s sprite sheet texture!\n", image.c_str());
232     }
233 }
234
235 void TextureRepository::_setSpellImage(TextureID TextureID, std::string^ spellI
mage,
236     int width, int height, int xOffset, i
nt yOffset) {
237     try {
238         ColorKey_t key = {0, 0, 0};
239         textures.emplace(TextureID, renderer);
240         Texture& texture = textures.at(TextureID);
241         texture.loadFromFile(spellImage, key, xOffset, yOffset);
242         _addSpellSprites(texture, 0, width, height);
243         _addSpellSprites(texture, height, width, height);
244         _addSpellSprites(texture, 2*height, width, height);
245         _addSpellSprites(texture, 3*height, width, height);
246     } catch (TPEException& e) {
247         throw TPEException("Failed to load %s sprite sheet texture!\n", spellImage.c_str());
248     }
249 }
250
251 void TextureRepository::_setTileImage(TextureID TextureID, std::string^ tileIma
ge, bool individualTile) {
252     try {
253         textures.emplace(TextureID, renderer);
254         Texture& texture = textures.at(TextureID);
255         texture.loadFromFile(tileImage);
256         _addTileSprites(texture, 0, individualTile);
257     } catch (TPEException& e) {
258         throw TPEException("Failed to load %s sprite sheet texture!\n", tileImage.c_str());

```

jul 21, 20 15:47

TextureRepository.cpp

Page 5/7

```

258     }
259 }
260
261 void TextureRepository::_setNPCImage(TextureID TextureID, std::string^ npcImage
, int width, int height
, int xOffset, int yOffset) {
262
263     try {
264         ColorKey_t key = {0, 0, 0};
265         textures.emplace(TextureID, renderer);
266         Texture& texture = textures.at(TextureID);
267         texture.loadFromFile(npcImage, key, xOffset, yOffset);
268         /*Front*/
269         _addNPCSprites(texture, 0, false, width, height);
270         /*Back*/
271         _addNPCSprites(texture, height, false, width, height);
272         /*Left*/
273         _addNPCSprites(texture, 2*height, true, width, height);
274         /*Rigth*/
275         _addNPCSprites(texture, 3*height, true, width, height);
276     } catch (TPEException& e) {
277         throw TPEException("Failed to load %s sprite sheet texture!\n", npcImage.c_str());
278     }
279 }
280
281 void TextureRepository::_addNPCSprites(Texture& texture, int y, bool lateralSide
, int width, int height) {
282     for (int i = 0; i < 5; ++i) {
283         texture.addSprite(width*i, y, width, height);
284     }
285     if (lateralSide) texture.addSprite(4*width, y, width, height);
286     else texture.addSprite(5*width, y, width, height);
287 }
288
289 void TextureRepository::_setBodyImage(TextureID TextureID, std::string^ bodyIma
ge) {
290     try {
291         ColorKey_t key = {0, 0, 0};
292         textures.emplace(TextureID, renderer);
293         Texture& texture = textures.at(TextureID);
294         texture.loadFromFile(bodyImage, key);
295         /*Front*/
296         _addBodySprites(texture, 0, false);
297         /*Back*/
298         _addBodySprites(texture, 45, false);
299         /*Left*/
300         _addBodySprites(texture, 90, true);
301         /*Rigth*/
302         _addBodySprites(texture, 135, true);
303     } catch (TPEException& e) {
304         throw TPEException("Failed to load %s sprite sheet texture!\n", bodyImage.c_str());
305     }
306 }
307
308 void TextureRepository::_setWeaponImage(TextureID TextureID, std::string^ weapo
nImage) {
309     try {
310         ColorKey_t key = {0, 0, 0};
311         textures.emplace(TextureID, renderer);
312         Texture& texture = textures.at(TextureID);
313         texture.loadFromFile(weaponImage, key);
314         /*Front*/
315         _addWeaponSprites(texture, 0, false);
316         /*Back*/
317         _addWeaponSprites(texture, 45, false);
318         /*Left*/
319         _addWeaponSprites(texture, 90, true);

```

jul 21, 20 15:47

TextureRepository.cpp

Page 6/7

```

320         /*Rigth*/
321         _addWeaponSprites(texture, 135, true);
322     } catch (TPEException& e) {
323         throw TPEException("Failed to load %s sprite sheet texture!\n", weaponImage.c_str());
324     }
325 }
326
327 void TextureRepository::_addWeaponSprites(Texture& texture, int y, bool lateralS
ide) {
328     texture.addSprite(0, y, 24, 45);
329     texture.addSprite(25, y, 25, 45);
330     texture.addSprite(51, y - 1, 23, 45);
331     texture.addSprite(76, y - 1, 23, 45);
332     texture.addSprite(101, y - 1, 24, 45);
333     if (lateralSide) texture.addSprite(101, y, 24, 45);
334     else texture.addSprite(126, y, 25, 45);
335 }
336
337 void TextureRepository::_addBodySprites(Texture& texture, int y, bool lateralSid
e) {
338     texture.addSprite(0, y, 24, 45); /*hasta 24 porque sino en la plate armor ha
y un poco de la otra imagen*/
339     texture.addSprite(25, y, 25, 45);
340     texture.addSprite(51, y, 24, 45); /*pongo 51 porque sino se veia un poco del
pie de otro en algunas ropas*/
341     texture.addSprite(75, y, 25, 45);
342     texture.addSprite(100, y, 25, 45);
343     if (lateralSide) texture.addSprite(100, y, 25, 45);
344     else texture.addSprite(125, y, 25, 45);
345 }
346
347 void TextureRepository::_setHeadImage(TextureID TextureID, std::string^ headIma
ge) {
348     try {
349         ColorKey_t key = {0, 0, 0};
350         textures.emplace(TextureID, renderer);
351         Texture& texture = textures.at(TextureID);
352         texture.loadFromFile(headImage, key);
353         texture.addSprite(0, 0, 17, 15);
354         texture.addSprite(17, 0, 17, 15);
355         texture.addSprite(34, 0, 17, 15);
356         texture.addSprite(51, 0, 17, 15);
357     } catch (TPEException& e) {
358         throw TPEException("Failed to load %s sprite sheet texture!\n", headImage.c_str());
359     }
360 }
361
362 void TextureRepository::_setHelmetImage(TextureID TextureID, std::string^ helme
tImage,
363                                     int xOffset, int yOffset) {
364     try {
365         ColorKey_t key = {0, 0, 0};
366         textures.emplace(TextureID, renderer);
367         Texture& texture = textures.at(TextureID);
368         texture.loadFromFile(helmetImage, key, xOffset, yOffset);
369         texture.addSprite(0, 0, 17, 17);
370         texture.addSprite(17, 0, 17, 17);
371         texture.addSprite(34, 0, 17, 17);
372         texture.addSprite(51, 0, 17, 17);
373     } catch (TPEException& e) {
374         throw TPEException("Failed to load %s sprite sheet texture!\n", helmetImage.c_str());
375     }
376 }
377
378 void TextureRepository::_setShieldImage(TextureID TextureID, std::string^ shiel
dImage) {

```

jul 21, 20 15:47

TextureRepository.cpp

Page 7/7

```

379     try {
380         ColorKey_t key = {0, 0, 0};
381         textures.emplace(TextureID, renderer);
382         Texture& texture = textures.at(TextureID);
383         texture.loadFromFile(shieldImage, key);
384         /*Front*/
385         _addShieldSprites(texture, 0, false);
386         /*Back*/
387         _addShieldSprites(texture, 45, false);
388         /*Left*/
389         _addShieldSprites(texture, 90, true);
390         /*Rigth*/
391         _addShieldSprites(texture, 135, true);
392     } catch (TPEException& e) {
393         throw TPEException("Failed to load %s sprite sheet texture!\n", shieldImage.c_str());
394     }
395 }
396
397 void TextureRepository::_addShieldSprites(Texture& texture, int y, bool lateralSide) {
398     texture.addSprite(0, y, 25, 35);
399     texture.addSprite(26, y, 25, 35);
400     texture.addSprite(51, y, 24, 35);
401     texture.addSprite(76, y, 25, 35);
402     texture.addSprite(101, y, 24, 35);
403     if (lateralSide) texture.addSprite(101, y, 24, 35);
404     else texture.addSprite(126, y, 25, 35);
405 }
406
407 void TextureRepository::_addTileSprites(Texture& texture, int y, bool individualTile) {
408     texture.addSprite(0, 0, TILE_WIDTH, TILE_HEIGHT);
409     if (!individualTile) {
410         texture.addSprite(TILE_WIDTH, 0, TILE_WIDTH, TILE_HEIGHT);
411         texture.addSprite(2*TILE_WIDTH, 0, TILE_WIDTH, TILE_HEIGHT);
412         texture.addSprite(3*TILE_WIDTH, 0, TILE_WIDTH, TILE_HEIGHT);
413     }
414 }
415
416 void TextureRepository::_addSprites(Texture& texture, int width, int height) {
417     texture.addSprite(0, 0, width, height);
418 }
419
420 void TextureRepository::_addSpellSprites(Texture& texture, int y, int width, int height) {
421     for (int i = 0; i < 6; ++i) {
422         texture.addSprite(width*i, y, width, height);
423     }
424 }
425
426 Texture& TextureRepository::getTexture(TextureID texture) {
427     return textures.at(texture);
428 }
429
430 SDL_Renderer &TextureRepository::getRenderer() const {
431     return renderer;
432 }

```

jul 21, 20 15:47

TextureID.h

Page 1/2

```

1  //
2  // Created by marcos on 7/5/20.
3  //
4
5  #ifndef ARGENTUM_TEXTUREID_H
6  #define ARGENTUM_TEXTUREID_H
7
8  enum TextureID {
9      Nothing, /*Auxiliar, lo uso para el equipo del Player*/
10     PlayerGhost,
11     BlueTunic,
12     BlueTunicDrop,
13     CommonClothing,
14     CommonClothingDrop,
15     Hood,
16     HoodDrop,
17     IronHelmet,
18     IronHelmetDrop,
19     IronShield,
20     IronShieldDrop,
21     KingArmor,
22     KingArmorDrop,
23     LeatherArmor,
24     LeatherArmorDrop,
25     MagicHat,
26     MagicHatDrop,
27     PlateArmor,
28     PlateArmorDrop,
29     TurtleShield,
30     TurtleShieldDrop,
31     DwarfHead,
32     ElfHead,
33     GnomeHead,
34     HumanHead,
35     AshRod,
36     AshRodDrop,
37     Axe,
38     AxeDrop,
39     CompositeBow,
40     CompositeBowDrop,
41     SimpleArrow,
42     CompositeArrow,
43     ElvenFluteDrop,
44     LinkedStaff,
45     LinkedStaffDrop,
46     GnarledStaff,
47     GnarledStaffDrop,
48     LongSword,
49     LongSwordDrop,
50     SimpleBow,
51     SimpleBowDrop,
52     WarHammer,
53     WarHammerDrop,
54     HealthPotion,
55     ManaPotion,
56     Grass,
57     PrettyGrass,
58     DeadGrass,
59     PrettyRoad,
60     Sand,
61     Water,
62     DarkWater,
63     Skeleton,
64     Goblin,
65     Zombie,
66     Spider,

```

jul 21, 20 15:47	TextureID.h	Page 2/2
67	Priest,	
68	Trader,	
69	Banker,	
70	Guard,	
71	Tree,	
72	LongTree,	
73	FatTree,	
74	PalmTree,	
75	DeadTree,	
76	Bush,	
77	DeadBush,	
78	House1,	
79	House2,	
80	House3,	
81	SunkenColumn,	
82	SunkenShip,	
83	BoneGuy,	
84	BrokenRipStone,	
85	DeadGuy,	
86	VeryDeadGuy,	
87	HangedGuy,	
88	RipStone,	
89	Explosion,	
90	MagicArrow,	
91	MagicMissile,	
92	Heal,	
93	Gold,	
94	Background,	
95	MainMenu	
96	};	
97		
98	#endif //ARGENTUM_TEXTUREID_H	

jul 21, 20 15:47	Texture.h	Page 1/2
1	//	
2	// Created by marcos on 6/6/20.	
3	//	
4		
5	#ifndef ARGENTUM_TEXTURE_H	
6	#define ARGENTUM_TEXTURE_H	
7		
8	<i>/*Esta clase representa la textura cargada. La textura puede contener mas de</i>	
9	<i>* una imagen, permitiendo renderizar solo lo que el programador elija*/</i>	
10		
11	#include <SDL.h>	
12	#include <SDL_image.h>	
13	#include "../libs/TPEException.h"	
14	#include "../Graphics/Text/Font.h"	
15	#include <string>	
16	#include <vector>	
17		
18		
19	struct ColorKey_t {	
20	int red;	
21	int green;	
22	int blue;	
23	};	
24		
25	struct SpriteDimensions_t {	
26	int width;	
27	int height;	
28	};	
29		
30	class Texture {	
31	private:	
32	SDL_Renderer& renderer;	
33	SDL_Texture* mTexture;	
34	int mWidth;	
35	int mHeight;	
36	int xOffset;	
37	int yOffset;	
38	int defaultScale;	
39	std::vector<SDL_Rect> gSpriteClips; <i>/*Sprites de la textura*/</i>	
40		
41	public:	
42	Texture(SDL_Renderer& renderer);	
43	~Texture();	
44	Texture(const Texture&) = delete;	
45	Texture& operator=(const Texture&) = delete;	
46	Texture(Texture^ other) noexcept;	
47		
48	<i>/*Carga la imagen de path, ignorando el color recibido en key. Opcionalmente</i>	
49	<i>* se le puede setear un offset de renderizacion y una escala distinta a la</i>	
50	<i>* imagen*/</i>	
51	void loadFromFile(const std::string& path, ColorKey_t key = {-1, -1, -1},	
52	int xOffset = 0, int yOffset = 0, int scale = 1);	
53		
54	<i>/*Especifica una dimension (un clip) que representa un sprite de la textura*</i>	
55	<i>/</i>	
56	void addSprite(int x, int y, int width, int height);	
57	<i>/*Hago sobrecarga para poder pasar por parametro default a la escala de la t</i>	
58	<i>extura</i>	
59	<i>* cuando la cree*/</i>	
60	void render(int x, int y, int spritePosition = 0, double angle = 0);	
61	<i>/*Renderiza el sprite de la textura en la posicion, angulo y escala indicado</i>	
62	<i>s*/</i>	
63	void render(int x, int y, int spritePosition, double angle, int scale);	

jul 21, 20 15:47

Texture.h

Page 2/2

```

64  /*Retorna las dimensiones del sprite de la textura*/
65  SpriteDimensions_t getSpriteDimensions(int spritePosition = 0);
66
67  /*Crea una textura en base al texto recibido*/
68  void loadFromRenderedText(const std::string& text, SDL_Color textColor, TTF_
Font* font);
69
70
71  private:
72      //Deallocates texture
73      void _free();
74  };
75
76
77  #endif //ARGENTUM_TEXTURE_H

```

jul 21, 20 15:47

Texture.cpp

Page 1/3

```

1  //
2  // Created by marcos on 6/6/20.
3  //
4
5  #include "Texture.h"
6
7  Texture::Texture(SDL_Renderer& renderer) : renderer(renderer) {
8      mTexture = nullptr;
9      mWidth = 0;
10     mHeight = 0;
11     xOffset = 0;
12     yOffset = 0;
13     defaultScale = 1;
14 }
15
16 Texture::~Texture() {
17     _free();
18 }
19
20 void Texture::loadFromFile(const std::string& path, ColorKey_t key, int xOff, in
t yOff,
21                                     int scale) {
22     //Libero la textura anterior
23     _free();
24
25     //carga la imagen de path
26     SDL_Surface* loadedSurface = IMG_Load(path.c_str());
27     if (loadedSurface == nullptr) {
28         throw TPEException("Unable to load image %s! SDL_image Error: %s\n",
path.c_str(), IMG_GetError());
29     }
30     else {
31         if (key.red > -1 ^ key.green > -1 ^ key.blue > -1) {
32             SDL_SetColorKey(loadedSurface, SDL_TRUE,
SDL_MapRGB(loadedSurface->format, key.red, key.green
, key.blue));
33             /*Con esto aclaras que pixel hacer transparente*/
34         }
35
36         //Crea la textura
37         mTexture = SDL_CreateTextureFromSurface(&renderer, loadedSurface);
38         if (mTexture == nullptr) {
39             //Si falla libero la superficie
40             SDL_FreeSurface(loadedSurface);
41             throw TPEException("Unable to create texture from %s! "
"Graphics Error: %s\n", path.c_str(), SDL_GetError());
42         }
43         else {
44             mWidth = loadedSurface->w;
45             mHeight = loadedSurface->h;
46         }
47
48         //Libero la superficie
49         SDL_FreeSurface(loadedSurface);
50     }
51
52     xOffset = xOff;
53     yOffset = yOff;
54     defaultScale = scale;
55 }
56
57 void Texture::_free() {
58     if (mTexture != nullptr) {
59         SDL_DestroyTexture(mTexture);
60         mTexture = nullptr;
61         mWidth = 0;
62         mHeight = 0;
63     }
64 }

```

jul 21, 20 15:47

Texture.cpp

Page 2/3

```

65 }
66
67 void Texture::render(int x, int y, int spritePosition, double angle, int scale)
68 {
69     SDL_Rect renderQuad = {x + xOffset, y + yOffset, mWidth, mHeight};
70     SDL_Rect& clip = gSpriteClips.at(spritePosition);
71
72     //Setea las dimensiones del rectangulo a renderizar
73     renderQuad.w = clip.w*scale;
74     renderQuad.h = clip.h*scale;
75
76     //Renderiza
77     SDL_RenderCopyEx(&renderer, mTexture, &clip, &renderQuad, angle,
78                     nullptr, SDL_FLIP_NONE);
79 }
80 void Texture::addSprite(int x, int y, int width, int height) {
81     gSpriteClips.push_back({x, y, width, height});
82 }
83
84 Texture::Texture(Texture^ other) noexcept : renderer(other.renderer) {
85     mWidth = other.mWidth;
86     mHeight = other.mHeight;
87     other.mWidth = 0;
88     other.mHeight = 0;
89     xOffset = other.xOffset;
90     yOffset = other.yOffset;
91     other.xOffset = 0;
92     other.yOffset = 0;
93     defaultScale = other.defaultScale;
94     other.defaultScale = 1;
95     mTexture = other.mTexture;
96     other.mTexture = nullptr;
97     gSpriteClips = std::move(other.gSpriteClips);
98 }
99
100 SpriteDimensions_t Texture::getSpriteDimensions(int spritePosition) {
101     SDL_Rect& spriteDimensions = gSpriteClips.at(spritePosition);
102     SpriteDimensions_t dimensions = {spriteDimensions.w, spriteDimensions.h};
103     return dimensions;
104 }
105
106 void Texture::loadFromRenderedText(const std::string& text, SDL_Color
107                                   textColor, TTF_Font* font) {
108     //Libero la textura anterior
109     _free();
110
111     //Creo una superficie con el texto
112     SDL_Surface* textSurface = TTF_RenderText_Solid(font, text.c_str(), textColo
113 r);
114 if(textSurface == nullptr) {
115     throw TPException("Unable to _render text surface! SDL_ttf Error: "
116                       "%s\n", TTF_GetError());
117 } else {
118     //Crea la textura
119     mTexture = SDL_CreateTextureFromSurface(&renderer, textSurface);
120
121     if(mTexture == nullptr) {
122         //Si falla libera la superficie
123         SDL_FreeSurface(textSurface);
124         throw TPException("Unable to create texture from rendered text! "
125                           "%s\n", SDL_GetError());
126     } else {
127         mWidth = textSurface->w;
128         mHeight = textSurface->h;
129         gSpriteClips.assign(1, {0, 0, mWidth, mHeight});
130     }
131 }

```

jul 21, 20 15:47

Texture.cpp

Page 3/3

```

129 }
130
131 //Libero al superficie
132 SDL_FreeSurface(textSurface);
133 }
134 }
135
136 void Texture::render(int x, int y, int spritePosition, double angle) {
137     render(x, y, spritePosition, angle, defaultScale);
138 }

```


jul 21, 20 15:47

PlayerTexture.h

Page 1/1

```

1 //
2 // Created by marcos on 6/6/20.
3 //
4
5 #ifndef ARGENTUM_PLAYERTEXTURE_H
6 #define ARGENTUM_PLAYERTEXTURE_H
7
8
9 #include "EntityTexture.h"
10 #include "../libs/TPEException.h"
11 #include "TextureRepository.h"
12 #include "PlayerEquipment.h"
13 #include "../libs/GameEnums.h"
14
15 /*Representa la textura de un player*/
16
17 class PlayerTexture : public EntityTexture {
18 private:
19     TextureRepository& textureRepo;
20     Texture* helmet;
21     Texture* head;
22     Texture* body;
23     Texture* shield;
24     Texture* weapon;
25     Font textFont;
26     Text nickname, level;
27     int textNicknameOffset{0}, textLevelOffset{0};
28
29 public:
30     PlayerTexture(TextureRepository& repo, PlayerEquipment equipment, const std::
:string& _level,
31                 const std::string& _nickname = "");
32
33     void renderFront(int x, int y, int frame) override;
34     void renderBack(int x, int y, int frame) override;
35     void renderRight(int x, int y, int frame) override;
36     void renderLeft(int x, int y, int frame) override;
37
38     void setLevel(const std::string &_level);
39
40     /*Cambia una textura del player*/
41     void equip(GameType::EquipmentPlace place, TextureID equipment);
42 };
43
44 #endif //ARGENTUM_PLAYERTEXTURE_H

```

jul 21, 20 15:47

PlayerTexture.cpp

Page 1/2

```

1 //
2 // Created by marcos on 6/6/20.
3 //
4
5 #include "PlayerTexture.h"
6 #include "../Client/GameConstants.h"
7
8 PlayerTexture::PlayerTexture(TextureRepository& repo, PlayerEquipment equipment,
9                             const std::string& _level, const std::string& _nickn
ame) : textureRepo(repo),
10                                     textFont("/var/Argentum/Assets/Fonts/Raleway-Medium.ttf", 20),
11                                     nickname(textFont, repo.getRenderer(), _nickname),
12                                     level(textFont, repo.getRenderer(), "(" + _level + "
)") {
13
14     textLevelOffset = level.getTextTextureWidth();
15     textNicknameOffset = (nickname.getTextTextureWidth() + textLevelOffset)/2;
16     if (equipment.helmet != Nothing) helmet = &textureRepo.getTexture(equipment.h
elmet);
17     else helmet = nullptr;
18     head = &textureRepo.getTexture(equipment.head);
19     body = &textureRepo.getTexture(equipment.body);
20     if (equipment.shield != Nothing) shield = &textureRepo.getTexture(equipment.s
hield);
21     else shield = nullptr;
22     if (equipment.weapon != Nothing) weapon = &textureRepo.getTexture(equipment.w
eapon);
23     else weapon = nullptr;
24 }
25
26 void PlayerTexture::renderFront(int x, int y, int frame) {
27     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
28     EntityTexture::render(head, x + 45, y + 15, 0);
29     EntityTexture::render(body, x + 37, y + 30, frame);
30     EntityTexture::render(helmet, x + 45, y + 15, 0);
31     EntityTexture::render(shield, x + 52, y + 30, frame);
32     EntityTexture::render(weapon, x + 37, y + 15, frame);
33     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
34     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
35 }
36
37 void PlayerTexture::renderBack(int x, int y, int frame) {
38     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
39     EntityTexture::render(head, x + 45, y + 15, 3);
40     EntityTexture::render(weapon, x + 40, y + 20, frame + 6);
41     EntityTexture::render(shield, x + 37, y + 15, frame + 6);
42     EntityTexture::render(body, x + 37, y + 30, frame + 6);
43     EntityTexture::render(helmet, x + 45, y + 11, 3);
44     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
45     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
46 }
47
48 void PlayerTexture::renderRight(int x, int y, int frame) {
49     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
50     EntityTexture::render(head, x + 46, y + 15, 1);
51     EntityTexture::render(shield, x + 40, y + 30, frame + 18);
52     EntityTexture::render(body, x + 37, y + 30, frame + 18);
53     EntityTexture::render(helmet, x + 45, y + 11, 1);
54     EntityTexture::render(weapon, x + 37, y + 20, frame + 18);
55     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
56     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI

```

jul 21, 20 15:47

PlayerTexture.cpp

Page 2/2

```

    LE_HEIGHT - 15);
}
57
58
59 void PlayerTexture::renderLeft(int x, int y, int frame) {
60     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
61     EntityTexture::render(head, x + 43, y + 15, 2);
62     EntityTexture::render(weapon, x + 33, y + 20, frame + 12);
63     EntityTexture::render(body, x + 37, y + 30, frame + 12);
64     EntityTexture::render(helmet, x + 43, y + 11, 2);
65     EntityTexture::render(shield, x + 47, y + 30, frame + 12);
66     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
67     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
68 }
69
70 void PlayerTexture::setLevel(const std::string &_level) {
71     *(level.updateText( "(" + _level + "));
72     textLevelOffset = level.getTextTextureWidth();
73     textNicknameOffset = (nickname.getTextTextureWidth() + textLevelOffset)/2;
74 }
75
76 void PlayerTexture::equip(GameType::EquipmentPlace place, TextureID equipment) {
77     Texture* texture = nullptr;
78     if (equipment ≠ Nothing) {
79         texture = &textureRepo.getTexture(equipment);
80     }
81     switch (place) {
82         case GameType::EQUIPMENT_PLACE_HEAD:
83             helmet = texture;
84             break;
85         case GameType::EQUIPMENT_PLACE_CHEST:
86             body = texture;
87             break;
88         case GameType::EQUIPMENT_PLACE_SHIELD:
89             shield = texture;
90             break;
91         case GameType::EQUIPMENT_PLACE_WEAPON:
92             weapon = texture;
93             break;
94         default:
95             //do nothing
96             break;
97     }
98 }
99

```

jul 21, 20 15:47

PlayerEquipment.h

Page 1/1

```

1 //
2 // Created by marcos on 6/27/20.
3 //
4
5 #ifndef ARGENTUM_PLAYEREQUIPMENT_H
6 #define ARGENTUM_PLAYEREQUIPMENT_H
7
8 #include "TextureID.h"
9
10 /*Encapsula lo equipado de un player*/
11
12 struct PlayerEquipment {
13     TextureID helmet;
14     TextureID head;
15     TextureID body;
16     TextureID shield;
17     TextureID weapon;
18 };
19
20 #endif //ARGENTUM_PLAYEREQUIPMENT_H

```

jul 21, 20 15:47

NPCTexture.h

Page 1/1

```

1 //
2 // Created by marcos on 6/8/20.
3 //
4
5 #ifndef ARGENTUM_NPCTEXTURE_H
6 #define ARGENTUM_NPCTEXTURE_H
7
8 #include "EntityTexture.h"
9 #include "../libs/TPEException.h"
10 #include "TextureRepository.h"
11 #include "../Graphics/Text/Text.h"
12
13 /*Representa la textura de un npc*/
14
15 class NPCTexture : public EntityTexture {
16 private:
17     TextureRepository& textureRepo;
18     Font textFont;
19     Texture* body;
20     Text nickname, level;
21     int textNicknameOffset{0}, textLevelOffset{0};
22
23 public:
24     explicit NPCTexture(TextureRepository& repo, TextureID texture, const std::s
tring& _level = "",
25                        const std::string& _nickname = "");
26     void renderFront(int x, int y, int frame) override;
27     void renderBack(int x, int y, int frame) override;
28     void renderRight(int x, int y, int frame) override;
29     void renderLeft(int x, int y, int frame) override;
30     void setLevel(const std::string& _level);
31 };
32
33
34 #endif //ARGENTUM_NPCTEXTURE_H

```

jul 21, 20 15:47

NPCTexture.cpp

Page 1/1

```

1 //
2 // Created by marcos on 6/8/20.
3 //
4
5 #include "NPCTexture.h"
6 #include "../Client/GameConstants.h"
7
8 NPCTexture::NPCTexture(TextureRepository& repo, TextureID texture, const std::st
ring& _level,
9                        const std::string& _nickname) : textureRepo(repo),
10                textFont("/var/Argentum/Assets/Fonts/Raleway-Medium.ttf", 20),
11                nickname(textFont, repo.getRenderer(), _nickname),
12                level(textFont, repo.getRenderer(), _level) {
13
14     textLevelOffset = level.getTextTextureWidth();
15     textNicknameOffset = (nickname.getTextTextureWidth() + textLevelOffset)/2;
16     if (textLevelOffset == 0) {
17         textLevelOffset = TILE_WIDTH/2 - 30;
18     }
19     body = &textureRepo.getTexture(texture);
20 }
21
22 void NPCTexture::renderFront(int x, int y, int frame) {
23     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
24     EntityTexture::render(body, x + 4, y - 20, frame);
25     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
26     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
27 }
28
29 void NPCTexture::renderBack(int x, int y, int frame) {
30     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
31     EntityTexture::render(body, x + 4, y - 20, frame + 6);
32     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
33     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
34 }
35
36 void NPCTexture::renderRight(int x, int y, int frame) {
37     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
38     EntityTexture::render(body, x + 4, y - 20, frame + 18);
39     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
40     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
41 }
42
43 void NPCTexture::renderLeft(int x, int y, int frame) {
44     if (frame < 0 ∨ frame > 5) throw TPEException("I dont have that character frame!");
45     EntityTexture::render(body, x + 4, y - 20, frame + 12);
46     nickname.render(x + TILE_WIDTH/2 - textNicknameOffset, y + TILE_HEIGHT - 15)
;
47     level.render(x + TILE_WIDTH/2 + textNicknameOffset - textLevelOffset, y + TI
LE_HEIGHT - 15);
48 }
49
50 void NPCTexture::setLevel(const std::string &_level) {
51     *(level.updateText(_level));
52     textLevelOffset = level.getTextTextureWidth();
53     textNicknameOffset = (nickname.getTextTextureWidth() + textLevelOffset)/2;
54 }
55

```

jul 21, 20 15:47

EntityTexture.h

Page 1/1

```

1  //
2  // Created by marcos on 6/9/20.
3  //
4
5  #ifndef ARGENTUM_ENTITYTEXTURE_H
6  #define ARGENTUM_ENTITYTEXTURE_H
7
8  #include "Texture.h"
9  #include "../Graphics/Text/Text.h"
10
11 /*Clase Abstracta, los hijos deben implementar la renderizacion*/
12
13 class EntityTexture {
14 public:
15     virtual void renderFront(int x, int y, int frame) = 0;
16     virtual void renderBack(int x, int y, int frame) = 0;
17     virtual void renderRight(int x, int y, int frame) = 0;
18     virtual void renderLeft(int x, int y, int frame) = 0;
19     static void render(Texture* texture, int x, int y, int spritePosition);
20     virtual ~EntityTexture() = default;
21 };
22
23
24 #endif //ARGENTUM_ENTITYTEXTURE_H

```

jul 21, 20 15:47

EntityTexture.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/9/20.
3  //
4
5  #include "EntityTexture.h"
6
7  const int SCALE = 2; /*Factor de escala de la imagen*/
8
9  void EntityTexture::render(Texture* texture, int x, int y, int spritePosition) {
10     if (texture != nullptr) texture->render(x, y, spritePosition, 0, SCALE);
11 }

```

jul 21, 20 15:47

SoundRepository.h

Page 1/1

```

1  #ifndef ARGENTUM_SOUNDREPOSITORY_H
2  #define ARGENTUM_SOUNDREPOSITORY_H
3
4
5  #include <iostream>
6  #include <unordered_map>
7  #include <queue>
8  #include "../libs/TPEException.h"
9  #include "Sound.h"
10
11 enum SoundID {SwordAttackSound, ArrowSound, Explotion1Sound, Explotion2Sound, Ex
plotion3Sound,
12     Death1Sound, Death2Sound, Attack1Sound, Attack2Sound, HeavyAttackSound,
StepSound, HealingSound,
13     ZombieSound, SpiderSound, SkeletonSound, GoblinSound, LevelUpSound};
14
15 class SoundRepository {
16 private:
17     std::unordered_map<SoundID, Sound> sounds;
18     Mix_Music* music{};
19 public:
20     SoundRepository();
21
22     /* Me devuelve la musica */
23     Mix_Music* getMusic();
24
25     /*Me devuelve el sonido */
26     Mix_Chunk* getSound(SoundID id);
27
28     ~SoundRepository();
29
30 private:
31     void _loadSounds();
32     void _loadMusic();
33 };
34
35 #endif //ARGENTUM_SOUNDREPOSITORY_H

```

jul 21, 20 15:47

SoundRepository.cpp

Page 1/2

```

1  #include "SoundRepository.h"
2
3  #define QUEUE_SIZE 3
4
5  #define SWORD_ATTACK_PATH "/var/Argentum/Assets/Sounds/swordAttack.wav"
6  #define HEAVY_ATTACK_PATH "/var/Argentum/Assets/Sounds/HeavyAttack.wav"
7  #define ATTACK_1_PATH "/var/Argentum/Assets/Sounds/genericAttack1.wav"
8  #define ATTACK_2_PATH "/var/Argentum/Assets/Sounds/genericAttack2.wav"
9  #define ARROW_PATH "/var/Argentum/Assets/Sounds/arrow.wav"
10 #define DEATH_1_PATH "/var/Argentum/Assets/Sounds/Death.wav"
11 #define DEATH_2_PATH "/var/Argentum/Assets/Sounds/YodaDeath.wav"
12 #define STEP_PATH "/var/Argentum/Assets/Sounds/Step.wav"
13 #define EXPLOSION_1_PATH "/var/Argentum/Assets/Sounds/Explotion1.wav"
14 #define EXPLOSION_2_PATH "/var/Argentum/Assets/Sounds/Explotion2.wav"
15 #define EXPLOSION_3_PATH "/var/Argentum/Assets/Sounds/Explotion3.wav"
16 #define HEALING_PATH "/var/Argentum/Assets/Sounds/heal.wav"
17 #define ZOMBIE_PATH "/var/Argentum/Assets/Sounds/Zombie.wav"
18 #define SPIDER_PATH "/var/Argentum/Assets/Sounds/Spider.wav"
19 #define SKELETON_PATH "/var/Argentum/Assets/Sounds/Skeleton.wav"
20 #define GOBLIN_PATH "/var/Argentum/Assets/Sounds/Goblin.wav"
21 #define LEVEL_UP_SOUND "/var/Argentum/Assets/Sounds/LevelUp.wav"
22
23 #define MUSIC_PATH "/var/Argentum/Assets/Sounds/argentumOnlineOST.mp3"
24
25 SoundRepository::SoundRepository() {
26     _loadSounds();
27     _loadMusic();
28 }
29
30 void SoundRepository::_loadSounds() {
31     try {
32         sounds.emplace(SwordAttackSound, SWORD_ATTACK_PATH);
33         sounds.emplace(HeavyAttackSound, HEAVY_ATTACK_PATH);
34         sounds.emplace(Attack1Sound, ATTACK_1_PATH);
35         sounds.emplace(Attack2Sound, ATTACK_2_PATH);
36         sounds.emplace(ArrowSound, ARROW_PATH);
37         sounds.emplace(Death1Sound, DEATH_1_PATH);
38         sounds.emplace(Death2Sound, DEATH_2_PATH);
39         sounds.emplace(StepSound, STEP_PATH);
40         sounds.emplace(Explotion1Sound, EXPLOSION_1_PATH);
41         sounds.emplace(Explotion2Sound, EXPLOSION_2_PATH);
42         sounds.emplace(Explotion3Sound, EXPLOSION_3_PATH);
43         sounds.emplace(HealingSound, HEALING_PATH);
44         sounds.emplace(ZombieSound, ZOMBIE_PATH);
45         sounds.emplace(SpiderSound, SPIDER_PATH);
46         sounds.emplace(SkeletonSound, SKELETON_PATH);
47         sounds.emplace(GoblinSound, GOBLIN_PATH);
48         sounds.emplace(LevelUpSound, LEVEL_UP_SOUND);
49
50     } catch (std::exception& e) {
51         std::cerr << e.what() << std::endl;
52     }
53     Mix_Volume(-1, 25);
54 }
55
56 void SoundRepository::_loadMusic(){
57     music = Mix_LoadMUS(MUSIC_PATH);
58     Mix_VolumeMusic(20);
59     if(music == nullptr) {
60         throw TPEException("Failed to load beat music! SDL_mixer Error: "
61                             "%s\n", Mix_GetError());
62     }
63 }
64
65 SoundRepository::~SoundRepository() {
66     //Cierra el mixer

```

jul 21, 20 15:47

SoundRepository.cpp

Page 2/2

```

67     Mix_FreeMusic(music);
68 }
69
70 Mix_Music* SoundRepository::getMusic() {
71     return music;
72 }
73
74 Mix_Chunk* SoundRepository::getSound(SoundID id) {
75     return sounds.at(id).getSound();
76 }

```

jul 21, 20 15:47

SoundPlayer.h

Page 1/1

```

1  //
2  // Created by ivan on 22/6/20.
3  //
4
5  #ifndef ARGENTUM_SOUNDPLAYER_H
6  #define ARGENTUM_SOUNDPLAYER_H
7
8  #include "SoundRepository.h"
9  #include <mutex>
10 #include "../libs/Timer.h"
11
12 class SoundPlayer {
13 private:
14     std::queue<SoundID> soundQueue;
15     SoundRepository repo;
16     std::mutex m;
17     Timer timer;
18     bool blocked{false}; /*Para regular que no le puedan meter muchos sonidos en
19 poco tiempo*/
20 public:
21     SoundPlayer();
22
23     /* Encola un sonido */
24     void queueSound(SoundID id);
25
26     /* Reproduce los sonidos que estan encolados */
27     void playSounds();
28
29     /* Reproduce la musica */
30     void playMusic();
31
32     /* Pausa la musica */
33     void pauseMusic();
34
35     /* Devuelve true si la musica se esta reproduciendo */
36     static bool isMusicPlaying();
37
38 private:
39     static SoundID _getRandomDeathSound();
40 };
41
42
43 #endif //ARGENTUM_SOUNDPLAYER_H

```

jul 21, 20 15:47

SoundPlayer.cpp

Page 1/2

```

1  //
2  // Created by ivan on 22/6/20.
3  //
4
5  #include "SoundPlayer.h"
6
7  #define QUEUE_SIZE 10
8
9  const int TIME_BETWEEN_SOUND_UPDATES = 50;
10
11 SoundPlayer::SoundPlayer() {
12     std::srand(std::clock());
13     timer.start();
14 }
15
16 void SoundPlayer::playMusic() {
17     std::lock_guard<std::mutex> l(m);
18     if( Mix_PlayingMusic() == 0 ) { //Empieza musica si no habia
19         Mix_PlayMusic(repo.getMusic(), -1);
20     } else if (Mix_PausedMusic() == 1) { //Resume musica si estaba en pausa
21         Mix_ResumeMusic();
22     }
23 }
24
25 void SoundPlayer::pauseMusic() {
26     std::lock_guard<std::mutex> l(m);
27     Mix_PauseMusic();
28 }
29
30 bool SoundPlayer::isMusicPlaying() {
31     return !Mix_PausedMusic();
32 }
33
34 SoundID SoundPlayer::_getRandomDeathSound() {
35     int rand = std::rand() % 2;
36     switch (rand) {
37         case 0:
38             return Death1Sound;
39         case 1:
40             return Death2Sound;
41         default:
42             return Death1Sound;
43     }
44 }
45
46 void SoundPlayer::queueSound(SoundID id) {
47     std::lock_guard<std::mutex> l(m);
48     if (timer.getTime() > TIME_BETWEEN_SOUND_UPDATES) {
49         blocked = false;
50     }
51     if (!blocked) {
52         if (id == Death1Sound) id = _getRandomDeathSound();
53         if (soundQueue.size() < QUEUE_SIZE)
54             soundQueue.push(id);
55     }
56 }
57
58 void SoundPlayer::playSounds() {
59     std::lock_guard<std::mutex> l(m);
60     Mix_Chunk* soundToPlay;
61     long unsigned int queueSize = soundQueue.size();
62     for (long unsigned int i = 0; i < queueSize; i++){
63         soundToPlay = repo.getSound(soundQueue.front());
64         Mix_PlayChannel(-1, soundToPlay, 0);
65         soundQueue.pop();
66     }

```

jul 21, 20 15:47

SoundPlayer.cpp

Page 2/2

```

67     if (!blocked) {
68         blocked = true;
69         timer.start();
70     }
71 }

```

jul 21, 20 15:47

Sound.h

Page 1/1

```

1  //
2  // Created by ivan on 10/6/20.
3  //
4
5  #ifndef ARGENTUM_SOUND_H
6  #define ARGENTUM_SOUND_H
7
8  #include <SDL_mixer.h>
9  #include <string>
10
11 #include "../libs/TPEException.h"
12
13 class Sound {
14 private:
15     Mix_Chunk* sound;
16 public:
17     explicit Sound(const std::string& path);
18     Sound(const Sound&) = delete;
19     Sound& operator=(const Sound&) = delete;
20
21     /* Devuelve el sonido */
22     Mix_Chunk* getSound();
23
24
25     ~Sound();
26 };
27
28
29 #endif // ARGENTUM_SOUND_H

```

jul 21, 20 15:47

Sound.cpp

Page 1/1

```

1  //
2  // Created by ivan on 10/6/20.
3  //
4
5  #include "Sound.h"
6
7  Sound::Sound(const std::string& path) {
8      sound = Mix_LoadWAV(path.c_str());
9      if(sound == nullptr) {
10         throw TPEException("Failed to load sound effect! SDL_mixer "
11                             "Error: %s\n", Mix_GetError());
12     }
13 }
14
15 Mix_Chunk *Sound::getSound() {
16     return sound;
17 }
18
19 Sound::~~Sound() {
20     Mix_FreeChunk(sound);
21 }
22
23

```


jul 21, 20 15:47

Window.h

Page 1/1

```

1 //
2 // Created by marcos on 11/6/20.
3 //
4
5 #ifndef ARGENTUM_WINDOW_H
6 #define ARGENTUM_WINDOW_H
7
8 #include <SDL.h>
9 #include <unordered_map>
10
11 enum Viewports {
12     ScreenViewport,
13     MapViewport,
14     InventoryViewport,
15     MinichatViewport,
16     PlayerInfoViewport
17 };
18
19 /*Esta clase maneja la instancia de la ventana del juego, tiene el ownership
20 * del renderizador y se encarga de mostrar los cambios renderizados y
21 * setear los viewports*/
22
23 class Window {
24 private:
25     SDL_Window* mWindow;
26     SDL_Renderer* renderer;
27
28     std::unordered_map<Viewports, SDL_Rect> viewports;
29     int mWidth;
30     int mHeight;
31
32     bool mFullScreen;
33     bool mMinimized;
34
35 public:
36     Window();
37
38     /* Maneja los eventos de la ventana. Por ejemplo resize o minimizar */
39     bool handleEvent(SDL_Event& e);
40
41     /* Limpia la ventana para poder renderizar */
42     void clear();
43
44     /* Muestra lo que haya renderizado */
45     void show();
46
47     /* Setea un viewport */
48     void setViewport(Viewports viewport);
49
50     SDL_Renderer& getRenderer();
51     int getWidth() const;
52     int getHeight() const;
53
54     ~Window();
55
56 private:
57     void _createViewports();
58     void _createWindow();
59     void _createRenderer();
60     void _handleResizeEvent(SDL_Event& e, bool& handled);
61 };
62
63 #endif //ARGENTUM_WINDOW_H

```

jul 21, 20 15:47

Window.cpp

Page 1/3

```

1 //
2 // Created by marcos on 11/6/20.
3 //
4
5 #include "Window.h"
6 #include "../Client/GameConstants.h"
7 #include "../libs/TPEException.h"
8 #include <SDL.h>
9
10 Window::Window() {
11     mWindow = nullptr;
12     renderer = nullptr;
13     mFullScreen = false;
14     mMinimized = false;
15     mWidth = 0;
16     mHeight = 0;
17     _createWindow();
18     _createRenderer();
19     _createViewports();
20 }
21
22 void Window::_createViewports(){
23     viewports.emplace(ScreenViewport, SDL_Rect{0,0,DEFAULT_SCREEN_WIDTH,
24                                                 DEFAULT_SCREEN_HEIGHT});
25     viewports.emplace(MapViewport, SDL_Rect{20,236,DEFAULT_MAP_WIDTH,
26                                                 DEFAULT_MAP_HEIGHT});
27
28     viewports.emplace(InventoryViewport, SDL_Rect{20 + DEFAULT_MAP_WIDTH,0,
29                                                 DEFAULT_INVENTORY_WIDTH,
30                                                 DEFAULT_INVENTORY_HEIGHT});
31
32     viewports.emplace(MinichatViewport, SDL_Rect{15 ,15,
33                                                 DEFAULT_MINICHAT_WIDTH,
34                                                 DEFAULT_MINICHAT_HEIGHT});
35
36     viewports.emplace(PlayerInfoViewport, SDL_Rect{20, DEFAULT_MINICHAT_HEIGHT
37 + DEFAULT_MAP_HEIGHT + 30, DEFAULT_PLAYER_INFO_WIDTH,
38 DEFAULT_PLAYER_INFO_HEIGHT});
39 }
40
41 void Window::_createWindow() {
42     mWindow = SDL_CreateWindow( "Argentum Online", SDL_WINDOWPOS_UNDEFINED,
43                                SDL_WINDOWPOS_UNDEFINED, DEFAULT_SCREEN_WIDTH,
44                                DEFAULT_SCREEN_HEIGHT,
45                                SDL_WINDOW_SHOWN | SDL_WINDOW_RESIZABLE);
46     if (mWindow != nullptr) {
47         mWidth = DEFAULT_SCREEN_WIDTH;
48         mHeight = DEFAULT_SCREEN_HEIGHT;
49     } else {
50         throw TPEException( "Window could not be created! Graphics Error: %s\n",
51                             SDL_GetError());
52     }
53 }
54
55 void Window::_createRenderer() {
56     renderer = SDL_CreateRenderer(mWindow, -1, SDL_RENDERER_ACCELERATED
57 | SDL_RENDERER_PRESENTVSYNC);
58     if (renderer == nullptr) throw TPEException( "Renderer could not be created! "
59 "Graphics Error: %s\n", SDL_GetError());
60 }
61
62 void Window::setRenderDrawColor(SDL_Renderer* renderer, 0xFF, 0xFF, 0xFF, 0xFF);
63
64 bool Window::handleEvent(SDL_Event& e) {
65     bool handled = false;
66     if (e.type == SDL_WINDOWEVENT) {

```

jul 21, 20 15:47

Window.cpp

Page 2/3

```

65     switch (e.window.event) {
66         case SDL_WINDOWEVENT_SIZE_CHANGED:
67             mWidth = e.window.data1;
68             mHeight = e.window.data2;
69             show();
70             break;
71         case SDL_WINDOWEVENT_EXPOSED:
72             SDL_RenderPresent(renderer);
73             break;
74         case SDL_WINDOWEVENT_FOCUS_GAINED:
75             mMinimized = false;
76             break;
77         case SDL_WINDOWEVENT_MAXIMIZED:
78             mMinimized = false;
79             break;
80         case SDL_WINDOWEVENT_RESTORED:
81             mMinimized = false;
82             break;
83     }
84     handled = true;
85 }
86 _handleResizeEvent(e, handled);
87 return handled;
88 }
89
90 void Window::_handleResizeEvent(SDL_Event& e, bool& handled) {
91     if (e.type == SDL_KEYDOWN ^ e.key.keysym.sym == SDLK_F1) {
92         handled = true;
93         mWidth = DEFAULT_SCREEN_WIDTH;
94         mHeight = DEFAULT_SCREEN_HEIGHT;
95         if (mFullScreen) {
96             SDL_SetWindowFullscreen(mWindow, SDL_FALSE);
97             SDL_SetWindowSize(mWindow, mWidth, mHeight);
98             mFullScreen = false;
99         } else {
100             SDL_SetWindowFullscreen(mWindow, SDL_TRUE);
101             mWidth = 1600;
102             mHeight = 1024;
103             SDL_SetWindowSize(mWindow, mWidth, mHeight);
104             mFullScreen = true;
105             mMinimized = false;
106         }
107     } else if (e.type == SDL_KEYDOWN ^ e.key.keysym.sym == SDLK_F2) {
108         handled = true;
109         SDL_SetWindowFullscreen(mWindow, SDL_FALSE);
110         mFullScreen = false;
111         mWidth = DEFAULT_SCREEN_WIDTH;
112         mHeight = DEFAULT_SCREEN_HEIGHT;
113         SDL_RestoreWindow(mWindow);
114         SDL_SetWindowSize(mWindow, mWidth, mHeight);
115     }
116 }
117
118 SDL_Renderer& Window::getRenderer() {
119     return *renderer;
120 }
121
122 Window::~Window() {
123     if (renderer != nullptr) SDL_DestroyRenderer(renderer);
124     if (mWindow != nullptr) SDL_DestroyWindow(mWindow);
125 }
126
127 void Window::clear() {
128     SDL_SetRenderDrawColor(renderer, 0xFF, 0xFF, 0xFF, 0xFF);
129     SDL_RenderClear(renderer);
130 }

```

jul 21, 20 15:47

Window.cpp

Page 3/3

```

131
132 void Window::show() {
133     if (!mMinimized) {
134         float x_scale = (float)mWidth/(float)DEFAULT_SCREEN_WIDTH;
135         float y_scale = (float)mHeight/(float)DEFAULT_SCREEN_HEIGHT;
136         SDL_RenderSetScale(renderer, x_scale, y_scale);
137         SDL_RenderPresent(renderer);
138     }
139 }
140
141 void Window::setViewport(Viewports viewport){
142     SDL_RenderSetViewport(renderer, &viewports.at(viewport));
143 }
144
145 int Window::getWidth() const {
146     return mWidth;
147 }
148
149 int Window::getHeight() const {
150     return mHeight;
151 }

```

jul 21, 20 15:47	MainMenu.h	Page 1/2
1	//	
2	// Created by ivan on 23/6/20.	
3	//	
4		
5	#ifndef ARGENTUM_MAINMENU_H	
6	#define ARGENTUM_MAINMENU_H	
7		
8	#include "../Graphics/Text/Text.h"	
9	#include "Window.h"	
10	#include "../libs/GameEnums.h"	
11		
12	class Socket;	
13	class GameInitializer;	
14		
15	struct GameStartInfo {	
16	GameType::Class myClass;	
17	GameType::Race myRace;	
18	};	
19		
20	class MainMenu {	
21	private:	
22	Window& window;	
23	Font mainMenuFont;	
24	Text text; <i>//El texto general. "Exit" "Connect" etc</i>	
25	Text hostInputText;	
26	Text portInputText;	
27	Text nicknameInputText;	
28	Text errorText;	
29	Text strength;	
30	Text constitution;	
31	Text intelligence;	
32	Text agility;	
33	Texture& mainMenuBackground;	
34	GameStartInfo info{};	
35	bool hostInput, portInput, nicknameInput;	
36		
37	public:	
38	MainMenu(Texture& texture, Window& window);	
39		
40	<i>/* Menu Principal antes de iniciar el juego */</i>	
41	void menuScreen(bool& quit, GameInitializer& initializer, Socket& socket);	
42		
43		
44	~MainMenu();	
45		
46	private:	
47	void _playerSelectionScreen(bool &quit, bool& createPlayer, bool& loadPlayer	
48);	
49	void _connectScreen(bool &quit, bool& goBack, Socket& socket);	
50	void _playerCreationScreen(bool &quit, bool& goBack);	
51	void _playerLoadScreen(bool &quit, bool& goBack);	
52	void _attemptToConnect(Socket &socket, bool &finished);	
53	void _connectLoadedPlayer(GameInitializer &initializer, Socket &socket, bool	
54	&success);	
55	void _connectCreatedPlayer(GameInitializer& initializer, Socket& socket, boo	
56	l& success);	
57		
58	void _renderPlayerSelectionScreen();	
59	void _renderConnectScreen();	
60	void _renderCreatePlayerScreen();	
61	void _renderLoadPlayerScreen();	
62		
63	void _renderRace();	
64	void _renderClass();	

jul 21, 20 15:47	MainMenu.h	Page 2/2
64	void _updateWarriorSkills();	
65	void _updatePaladinSkills();	
66	void _updateClericSkills();	
67	void _updateWizardSkills();	
68	void _updateGnomeSkills();	
69	void _updateDwarfSkills();	
70	void _updateHumanSkills();	
71	void _updateElfSkills();	
72		
73	void _handleTextInput(SDL_Event &e);	
74	void _handleBackspace();	
75	void _verifyClassSelection(int x, int y);	
76	void _verifyRaceSelection(int x, int y);	
77	static bool _isInsideRect(int x, int y, SDL_Rect rect);	
78	};	
79		
80		
81	#endif //ARGENTUM_MAINMENU_H	

jul 21, 20 15:47

MainMenu.cpp

Page 1/10

```

1  //
2  // Created by ivan on 23/6/20.
3  //
4
5  #include "MainMenu.h"
6  #include "../Client/GameConstants.h"
7  #include "../Client/GameInitializer.h"
8  #include "../libs/Socket.h"
9  #include <netdb.h>
10
11 #define START_BUTTON {1375, 875, 100, 25}
12 #define CONNECT_BUTTON {1375, 875, 100, 25}
13 #define EXIT_BUTTON {50,875,90,25}
14 #define BACK_BUTTON {50,875,90,25}
15
16 #define INPUT_HOST_BOX {115,100,365,25}
17 #define INPUT_PORT_BOX {115,200,365,25}
18 #define INPUT_NICKNAME_BOX {165,100,365,25}
19
20 #define WARRIOR_BUTTON {150, 200, 100, 25}
21 #define WIZARD_BUTTON {300, 200, 100, 25}
22 #define CLERIC_BUTTON {450, 200, 100, 25}
23 #define PALADIN_BUTTON {600, 200, 100, 25}
24
25 #define HUMAN_BUTTON {150, 300, 100, 25}
26 #define ELF_BUTTON {300, 300, 100, 25}
27 #define DWARF_BUTTON {450, 300, 100, 25}
28 #define GNOME_BUTTON {600, 300, 100, 25}
29
30 #define LOAD_PLAYER_BUTTON {50,200,175,25}
31 #define CREATE_PLAYER_BUTTON {50,100,175,25}
32
33 #define MAX_TEXT_LEN 25
34 #define MAX_NICKNAME_LEN 13
35
36 #define MAIN_MENU_FONT_PATH "/var/Argentum/Assets/Fonts/medieval.ttf"
37
38
39 MainMenu::MainMenu(Texture& texture, Window& window) : window(window),
40 mainMenuFont(MAIN_MENU_FONT_PATH, 25),
41 text(mainMenuFont, window.getRenderer()),
42 hostInputText(mainMenuFont, window.getRenderer()),
43 portInputText(mainMenuFont, window.getRenderer()),
44 nicknameInputText(mainMenuFont, window.getRenderer()),
45 errorText(mainMenuFont, window.getRenderer()),
46 strength(mainMenuFont, window.getRenderer()),
47 constitution(mainMenuFont, window.getRenderer()),
48 intelligence(mainMenuFont, window.getRenderer()),
49 agility(mainMenuFont, window.getRenderer()),
50 mainMenuBackground(texture) {
51
52     hostInput = false;
53     portInput = false;
54     nicknameInput = false;
55     SDL_StartTextInput();
56     info = {GameType::WARRIOR, GameType::HUMAN};
57 }
58
59 void MainMenu::menuScreen(bool& quit, GameInitializer& initializer, Socket& socket) {
60     bool createPlayer = false;
61     bool loadPlayer = false;
62     bool success = false;
63     bool goBack = false;
64
65     while (¬success ^ ¬quit) {

```

jul 21, 20 15:47

MainMenu.cpp

Page 2/10

```

66     _playerSelectionScreen(quit, createPlayer, loadPlayer); //Veo si quiere h
acer load o create
67     if (createPlayer) {
68
69         do {
70             goBack = false;
71             _playerCreationScreen(quit, goBack);
72             if (goBack) break;
73             _connectScreen(quit, goBack, socket);
74             if (¬quit ^ ¬goBack)
75                 _connectCreatedPlayer(initializer, socket, success);
76         } while (goBack);
77
78     } else if (loadPlayer) {
79
80         do {
81             goBack = false;
82             _playerLoadScreen(quit, goBack);
83             if (goBack) break;
84             _connectScreen(quit, goBack, socket);
85             if (¬quit ^ ¬goBack) //xq puedo hacer quit en el run
86                 _connectLoadedPlayer(initializer, socket, success);
87         } while (goBack);
88
89     }
90
91     //Chequeo goBack porque si no me trate de conectar no tengo que cerrar e
l socket
92     if (¬success ^ ¬goBack) {
93         socket.close();
94     }
95 }
96
97
98 /* Intenta conectarse al host/port que ingresa el usuario */
99 void MainMenu::_connectScreen(bool& quit, bool& goBack, Socket& socket) {
100     errorText.updateText("");
101     bool finished = quit;
102     SDL_Event e;
103     while (¬finished) {
104         while (SDL_PollEvent(&e) ≠ 0){
105
106             if (e.type == SDL_QUIT){
107                 quit = true;
108                 finished = true;
109             }
110
111             window.handleEvent(e);
112             if (e.type == SDL_MOUSEBUTTONDOWN){
113                 int x = 0, y = 0;
114                 SDL_GetMouseState(&x, &y);
115                 x = (float)x * ((float)DEFAULT_SCREEN_WIDTH / (float>window.getW
idth());
116                 y = (float)y * ((float)DEFAULT_SCREEN_HEIGHT / (float>window.get
Height());
117
118                 if (_isInsideRect(x, y, INPUT_HOST_BOX)) {
119                     hostInput = true;
120                     portInput = false;
121                 } else if (_isInsideRect(x, y, INPUT_PORT_BOX)) {
122                     hostInput = false;
123                     portInput = true;
124                 } else if (_isInsideRect(x,y,CONNECT_BUTTON)) {
125                     _attemptToConnect(socket, finished);
126                 } else if (_isInsideRect(x,y,BACK_BUTTON)) {
127                     goBack = true;
128                     finished = true;

```

jul 21, 20 15:47

MainMenu.cpp

Page 3/10

```

128     }
129     } else if (e.type == SDL_TEXTINPUT){
130         _handleTextInput(e);
131     } else if (e.type == SDL_KEYDOWN) {
132         if (e.key.keysym.sym == SDLK_BACKSPACE) {
133             _handleBackspace();
134         }
135     }
136 }
137
138 _renderConnectScreen();
139 }
140 hostInput = false;
141 portInput = false;
142 errorText.updateText("");
143 }
144
145 /* Chequea si el usuario quiere cargar un jugador o crear uno nuevo */
146 void MainMenu::_playerSelectionScreen(bool& quit, bool& createPlayer, bool& load
Player) {
147     bool finished = quit;
148     SDL_Event e;
149     while (!finished) {
150         while (SDL_PollEvent(&e) != 0) {
151
152             if (e.type == SDL_QUIT){
153                 quit = true;
154                 finished = true;
155             }
156             window.handleEvent(e);
157             if (e.type == SDL_MOUSEBUTTONDOWN) {
158                 int x = 0, y = 0;
159                 SDL_GetMouseState(&x, &y);
160                 x = (float) x * ((float) DEFAULT_SCREEN_WIDTH / (float) window.g
etWidth());
161                 y = (float) y * ((float) DEFAULT_SCREEN_HEIGHT / (float) window.
getHeight());
162                 if (_isInsideRect(x, y, EXIT_BUTTON)) {
163                     quit = true;
164                     finished = true;
165                 } else if (_isInsideRect(x, y, CREATE_PLAYER_BUTTON)) {
166                     createPlayer = true;
167                     loadPlayer = false;
168                     finished = true;
169                 } else if (_isInsideRect(x, y, LOAD_PLAYER_BUTTON)) {
170                     loadPlayer = true;
171                     createPlayer = false;
172                     finished = true;
173                 }
174             }
175         }
176         _renderPlayerSelectionScreen();
177     }
178 }
179
180 /* Permite al usuario elegir el nickname del player que quiere cargar */
181 void MainMenu::_playerLoadScreen(bool &quit, bool& goBack) {
182     errorText.updateText("");
183     bool finished = quit;
184     SDL_Event e;
185     while (!finished){
186         while (SDL_PollEvent(&e) != 0){
187             if (e.type == SDL_QUIT){
188                 quit = true;
189                 finished = true;

```

jul 21, 20 15:47

MainMenu.cpp

Page 4/10

```

191     }
192     //Por si hago resize
193     window.handleEvent(e);
194     if (e.type == SDL_MOUSEBUTTONDOWN){
195         int x = 0, y = 0;
196         SDL_GetMouseState( &x, &y );
197         x = (float)x * ((float)DEFAULT_SCREEN_WIDTH/(float>window.getWid
th());
198         y = (float)y * ((float)DEFAULT_SCREEN_HEIGHT/(float>window.getHe
ight());
199         if (_isInsideRect(x, y, INPUT_NICKNAME_BOX)){
200             nicknameInput = true;
201         } else if (_isInsideRect(x,y,BACK_BUTTON)) {
202             goBack = true;
203             finished = true;
204         } else if (_isInsideRect(x,y,START_BUTTON)) {
205             if (!nicknameInputText.getText().empty()) {
206                 finished = true;
207             } else {
208                 errorText.updateText( "Nickname is empty" );
209             }
210         }
211     } else if (e.type == SDL_TEXTINPUT){
212         _handleTextInput(e);
213     } else if (e.type == SDL_KEYDOWN) {
214         if (e.key.keysym.sym == SDLK_BACKSPACE) {
215             _handleBackspace();
216         }
217     }
218 }
219 _renderLoadPlayerScreen();
220 }
221 nicknameInput = false;
222 }
223
224 /* Permite al usuario elegir los datos del jugador que quiere crear */
225 void MainMenu::_playerCreationScreen(bool &quit, bool &goBack) {
226     errorText.updateText("");
227     bool finished = quit;
228     SDL_Event e;
229     while (!finished){
230         while (SDL_PollEvent(&e) != 0){
231             if (e.type == SDL_QUIT){
232                 quit = true;
233                 finished = true;
234             }
235             //Por si hago resize
236             window.handleEvent(e);
237             if (e.type == SDL_MOUSEBUTTONDOWN){
238                 int x = 0, y = 0;
239                 SDL_GetMouseState( &x, &y );
240                 x = (float)x * ((float)DEFAULT_SCREEN_WIDTH/(float>window.getWid
th());
241                 y = (float)y * ((float)DEFAULT_SCREEN_HEIGHT/(float>window.getHe
ight());
242                 if (_isInsideRect(x, y, INPUT_NICKNAME_BOX)){
243                     nicknameInput = true;
244                 } else if (_isInsideRect(x,y,BACK_BUTTON)) {
245                     goBack = true;
246                     finished = true;
247                 } else if (_isInsideRect(x,y,START_BUTTON)) {
248                     if (!nicknameInputText.getText().empty()) {
249                         finished = true;
250                     } else {
251                         errorText.updateText( "Nickname is empty" );
252                     }

```

jul 21, 20 15:47

MainMenu.cpp

Page 5/10

```

253         } else {
254             _verifyClassSelection(x, y);
255             _verifyRaceSelection(x, y);
256         }
257     } else if (e.type == SDL_TEXTINPUT){
258         _handleTextInput(e);
259     } else if (e.type == SDL_KEYDOWN) {
260         if (e.key.keysym.sym == SDLK_BACKSPACE) {
261             _handleBackspace();
262         }
263     }
264 }
265 _renderCreatePlayerScreen();
266 }
267 nicknameInput = false;
268 }
269
270 /* Verifica si se hizo click en alguna clase */
271 void MainMenu::_verifyClassSelection(int x, int y){
272     if (_isInsideRect(x,y,WARRIOR_BUTTON)) info.myClass = GameType::WARRIOR;
273     else if (_isInsideRect(x,y,WIZARD_BUTTON)) info.myClass = GameType::WIZARD;
274     else if (_isInsideRect(x,y,CLERIC_BUTTON)) info.myClass = GameType::CLERIC;
275     else if (_isInsideRect(x,y,PALADIN_BUTTON)) info.myClass = GameType::PALADIN;
276 }
277
278 /* Verifica si se hizo click en alguna raza */
279 void MainMenu::_verifyRaceSelection(int x, int y) {
280     if (_isInsideRect(x,y,HUMAN_BUTTON)) info.myRace = GameType::HUMAN;
281     else if (_isInsideRect(x,y,ELF_BUTTON)) info.myRace = GameType::ELF;
282     else if (_isInsideRect(x,y,DWARF_BUTTON)) info.myRace = GameType::DWARF;
283     else if (_isInsideRect(x,y,GNOME_BUTTON)) info.myRace = GameType::GNOME;
284 }
285
286 /* Intenta conectarse al servidor con el player que se quiere crear */
287 void MainMenu::_connectCreatedPlayer(GameInitializer& initializer, Socket& socket, bool& success) {
288     if (nicknameInputText.getText().find(' ') != std::string::npos) {
289         errorText.updateText("Nickname cannot contain spaces");
290         return;
291     }
292     if (!nicknameInputText.getText().empty()) {
293         initializer.createPlayer(nicknameInputText.getText(), info.myRace, info.myClass);
294         GameType::ConnectionResponse response{};
295         socket.receive(reinterpret_cast<char*>(&response), sizeof(response));
296         response = static_cast<GameType::ConnectionResponse>(ntohl(response));
297         switch (response) {
298             case GameType::ACCEPTED:
299                 success = true;
300                 break;
301             case GameType::UNAVAILABLE_PLAYER:
302                 errorText.updateText("Nickname\" + nicknameInputText.getText() +
303                                     "\" is already in use");
304                 break;
305             case GameType::UNKNOWN_SERVER_ERROR:
306                 errorText.updateText("Unknown Server Error");
307                 break;
308             default:
309                 errorText.updateText("Unknown Error");
310                 break;
311         }
312     }
313 }
314 }
315

```

jul 21, 20 15:47

MainMenu.cpp

Page 6/10

```

316 /* Intenta conectarse al servidor con el player que se quiere cargar */
317 void MainMenu::_connectLoadedPlayer(GameInitializer& initializer, Socket& socket, bool& success) {
318     if (!nicknameInputText.getText().empty()) {
319         initializer.loadPlayer(nicknameInputText.getText());
320         GameType::ConnectionResponse response{};
321         socket.receive(reinterpret_cast<char*>(&response), sizeof(int32_t));
322         response = static_cast<GameType::ConnectionResponse>(ntohl(response));
323         switch (response) {
324             case GameType::ACCEPTED:
325                 success = true;
326                 break;
327             case GameType::INEXISTENT_PLAYER:
328                 errorText.updateText("Player\" + nicknameInputText.getText() +
329                                     "\" does not exist");
330                 break;
331             case GameType::UNAVAILABLE_PLAYER:
332                 errorText.updateText("Player\" + nicknameInputText.getText() +
333                                     "\" is already logged in");
334                 break;
335             case GameType::UNKNOWN_SERVER_ERROR:
336                 errorText.updateText("Unknown Server Error");
337                 break;
338             default:
339                 errorText.updateText("Unknown Error");
340                 break;
341         }
342     }
343 }
344
345 /* Intenta establecer una conexion con el servidor */
346 void MainMenu::_attemptToConnect(Socket& socket, bool& finished) {
347     try {
348         socket.connect(hostInputText.getText(), portInputText.getText());
349         finished = true;
350     } catch (std::exception& e) {
351         errorText.updateText("Could not connect");
352     }
353 }
354
355 /* Chequea en donde se quiere hacer el input y lo procesa */
356 void MainMenu::_handleTextInput(SDL_Event& e) {
357     std::string newInput = e.text.text;
358     if (hostInput) {
359         if (hostInputText.getTextLength() < MAX_TEXT_LEN)
360             hostInputText += std::move(newInput);
361     } else if (portInput) {
362         if (portInputText.getTextLength() < MAX_TEXT_LEN)
363             portInputText += std::move(newInput);
364     } else if (nicknameInput) {
365         if (nicknameInputText.getTextLength() < MAX_NICKNAME_LEN)
366             nicknameInputText += std::move(newInput);
367     }
368 }
369
370 /* Borra una letra del texto donde se hizo click */
371 void MainMenu::_handleBackspace() {
372     if (hostInput) {
373         --hostInputText;
374     } else if (portInput) {
375         --portInputText;
376     } else if (nicknameInput) {
377         --nicknameInputText;
378     }
379 }

```

jul 21, 20 15:47

MainMenu.cpp

Page 7/10

```

380
381 /* Chequea si se hizo click dentro de un rectangulo */
382 bool MainMenu::_isInsideRect(int x, int y, SDL_Rect rect){
383     return ((x > rect.x) ^ (x < rect.x + rect.w) ^ (y > rect.y) ^ (y <
384         rect.y + rect.h));
385 }
386
387 /* Renderiza la pantalla de conexion */
388 void MainMenu::_renderConnectScreen(){
389     window.clear();
390     window.setViewport(ScreenViewport);
391     mainMenuBackground.render(0,0);
392
393     /* Outline de la text box para el input de host y port */
394     SDL_Rect outlineRect = INPUT_HOST_BOX;
395     SDL_SetRenderDrawColor(&window.getRenderer(), 0x00, 0x00,
396         0x00, 0xFF);
397     SDL_RenderDrawRect( &window.getRenderer(), &outlineRect );
398     outlineRect = INPUT_PORT_BOX;
399     SDL_SetRenderDrawColor(&window.getRenderer(), 0x00, 0x00,
400         0x00, 0xFF);
401     SDL_RenderDrawRect( &window.getRenderer(), &outlineRect );
402     text.updateText("Host: ").operator*({0, 0, 0});
403     text.render(50, 100);
404     text.updateText("Port: ").operator*({0, 0, 0});
405     text.render(50, 200);
406     *(text.updateText("Connect"));
407     text.render(1375, 875);
408     *(text.updateText("Back"));
409     text.render(50, 875);
410     hostInputText.operator*({0, 0, 0}).render(115, 100);
411     portInputText.operator*({0, 0, 0}).render(115, 200);
412     (*errorText).render(650, 875);
413     window.show();
414 }
415
416 void MainMenu::_renderPlayerSelectionScreen() {
417     window.clear();
418     window.setViewport(ScreenViewport);
419     mainMenuBackground.render(0,0);
420     text.updateText("Create Player").operator*({0, 0, 0});
421     text.render(50, 100);
422     text.updateText("Load Player").operator*({0, 0, 0});
423     text.render(50, 200);
424     *(text.updateText("Exit"));
425     text.render(50, 875);
426     (*errorText).render(650, 875);
427     window.show();
428 }
429
430 void MainMenu::_renderLoadPlayerScreen() {
431     window.clear();
432     window.setViewport(ScreenViewport);
433     mainMenuBackground.render(0,0);
434     /* Outline de la text box para el input de nickname */
435     SDL_Rect outlineRect = INPUT_NICKNAME_BOX;
436     SDL_SetRenderDrawColor(&window.getRenderer(), 0x00, 0x00,
437         0x00, 0xFF);
438     SDL_RenderDrawRect( &window.getRenderer(), &outlineRect );
439     text.updateText("Nickname: ").operator*({0, 0, 0});
440     text.render(50, 100);
441     *(text.updateText("Start"));
442     text.render(1375, 875);
443     *(text.updateText("Back"));
444     text.render(50, 875);
445     nicknameInputText.operator*({0, 0, 0}).render(165, 100);

```

jul 21, 20 15:47

MainMenu.cpp

Page 8/10

```

446     (*errorText).render(650, 875);
447     window.show();
448 }
449
450 void MainMenu::_renderCreatePlayerScreen() {
451     window.clear();
452     window.setViewport(ScreenViewport);
453     mainMenuBackground.render(0,0);
454     /* Outline de la text box para el input de nickname */
455     SDL_Rect outlineRect = INPUT_NICKNAME_BOX;
456     SDL_SetRenderDrawColor(&window.getRenderer(), 0x00, 0x00,
457         0x00, 0xFF);
458     SDL_RenderDrawRect( &window.getRenderer(), &outlineRect );
459     text.updateText("Nickname: ").operator*({0, 0, 0});
460     text.render(50, 100);
461     (strength.updateText("Strength")).operator*({0x00,0x00,0x00});
462     (constitution.updateText("Constitution")).operator*({0x00,0x00,0x00});
463     (intelligence.updateText("Intelligence")).operator*({0x00,0x00,0x00});
464     (agility.updateText("Agility")).operator*({0x00,0x00,0x00});
465     _renderClass();
466     _renderRace();
467     strength.render(125, 400);
468     constitution.render(125, 500);
469     intelligence.render(375, 400);
470     agility.render(375, 500);
471     nicknameInputText.operator*({0, 0, 0}).render(165, 100);
472     (*errorText).render(650, 875);
473     *(text.updateText("Start"));
474     text.render(1375, 875);
475     *(text.updateText("Back"));
476     text.render(50, 875);
477     window.show();
478 }
479
480 void MainMenu::_renderClass() {
481     text.updateText("Class: ").operator*({0x00,0x00,0x00});
482     text.render(50, 200);
483     text.updateText("Warrior").operator*({0x00,0x00,0x00});
484     text.render(150, 200);
485     text.updateText("Wizard").operator*({0x00,0x00,0x00});
486     text.render(300, 200);
487     text.updateText("Cleric").operator*({0x00,0x00,0x00});
488     text.render(450, 200);
489     text.updateText("Paladin").operator*({0x00,0x00,0x00});
490     text.render(600, 200);
491     /* Outline de la clase que tengo seleccionada */
492     SDL_Rect outlineRect;
493     switch (info.myClass) {
494         case GameType::WARRIOR:
495             outlineRect = WARRIOR_BUTTON;
496             _updateWarriorSkills();
497             break;
498         case GameType::WIZARD:
499             outlineRect = WIZARD_BUTTON;
500             _updateWizardSkills();
501             break;
502         case GameType::CLERIC:
503             outlineRect = CLERIC_BUTTON;
504             _updateClericSkills();
505             break;
506         case GameType::PALADIN:
507             outlineRect = PALADIN_BUTTON;
508             _updatePaladinSkills();
509             break;
510     }
511     SDL_SetRenderDrawColor(&window.getRenderer(), 0x3f, 0x2a,

```

jul 21, 20 15:47

MainMenu.cpp

Page 9/10

```

512         0x14, 0xFF);
513     SDL_RenderDrawRect( &window.getRenderer(), &outlineRect );
514 }
515
516 void MainMenu::_renderRace() {
517     text.updateText("Race: ").operator*({0x00,0x00,0x00});
518     text.render(50, 300);
519     text.updateText("Human").operator*({0x00,0x00,0x00});
520     text.render(150, 300);
521     text.updateText("Elf").operator*({0x00,0x00,0x00});
522     text.render(300, 300);
523     text.updateText("Dwarf").operator*({0x00,0x00,0x00});
524     text.render(450, 300);
525     text.updateText("Gnome").operator*({0x00,0x00,0x00});
526     text.render(600, 300);
527     /* Outline de la raza que tengo seleccionada */
528     SDL_Rect outlineRect;
529     switch (info.myRace) {
530         case GameType::HUMAN:
531             outlineRect = HUMAN_BUTTON;
532             _updateHumanSkills();
533             break;
534         case GameType::ELF:
535             outlineRect = ELF_BUTTON;
536             _updateElfSkills();
537             break;
538         case GameType::DWARF:
539             outlineRect = DWARF_BUTTON;
540             _updateDwarfSkills();
541             break;
542         case GameType::GNOME:
543             outlineRect = GNOME_BUTTON;
544             _updateGnomeSkills();
545             break;
546     }
547     SDL_SetRenderDrawColor(&window.getRenderer(), 0x3f, 0x2a,
548         0x14, 0xFF);
549     SDL_RenderDrawRect( &window.getRenderer(), &outlineRect );
550 }
551
552 void MainMenu::_updateWarriorSkills() {
553     (strength += "++++").operator*({0, 0, 0});
554     (constitution += "+++").operator*({0, 0, 0});
555 }
556
557 void MainMenu::_updateWizardSkills() {
558     (intelligence += "++++").operator*({0, 0, 0});
559 }
560
561 void MainMenu::_updateClericSkills() {
562     (strength += "++").operator*({0, 0, 0});
563     (constitution += "++").operator*({0, 0, 0});
564     (intelligence += "+++").operator*({0, 0, 0});
565 }
566
567 void MainMenu::_updatePaladinSkills() {
568     (strength += "++++").operator*({0, 0, 0});
569     (constitution += "++++").operator*({0, 0, 0});
570     (intelligence += "+").operator*({0, 0, 0});
571 }
572
573 void MainMenu::_updateHumanSkills() {
574     (strength += "++").operator*({0, 0, 0});
575     (constitution += "++").operator*({0, 0, 0});
576     (intelligence += "++").operator*({0, 0, 0});
577     (agility += "++").operator*({0, 0, 0});

```

jul 21, 20 15:47

MainMenu.cpp

Page 10/10

```

578 }
579
580 void MainMenu::_updateElfSkills() {
581     (intelligence += "++++").operator*({0, 0, 0});
582     (agility += "+++").operator*({0, 0, 0});
583 }
584
585 void MainMenu::_updateDwarfSkills() {
586     (strength += "++++").operator*({0, 0, 0});
587     (constitution += "+++").operator*({0, 0, 0});
588 }
589
590 void MainMenu::_updateGnomeSkills() {
591     (strength += "++").operator*({0, 0, 0});
592     (constitution += "+++").operator*({0, 0, 0});
593     (intelligence += "++").operator*({0, 0, 0});
594 }
595
596 MainMenu::~MainMenu() {
597     SDL_StopTextInput();
598 }

```


jul 21, 20 15:47

Spell.h

Page 1/1

```

1 //
2 // Created by marcos on 6/13/20.
3 //
4
5 #ifndef ARGENTUM_SPELL_H
6 #define ARGENTUM_SPELL_H
7
8 #include "../Texture/Texture.h"
9
10 class Spell {
11 private:
12     Texture& sTexture;
13     SDL_Rect& camera;
14     float timePassed;
15     int currentFrame;
16     float xPosition, width;
17     float yPosition, height;
18     bool finished{false};
19
20 public:
21     Spell(Texture& texture, SDL_Rect& camera, float x, float y);
22
23     /*Renderiza el hechizo (si fuera visible)*/
24     void render();
25
26     /*Actualiza el frame de la animacion del hechizo*/
27     void updateFrame(double timeStep);
28
29     /*Retorna true si el hechizo completo su animacion asi lo borro*/
30     bool finishedAnimation() const;
31
32     /*Cambia la posicion, lo utilizo para que vaya siguiendo al entity*/
33     void setPosition(float x, float y);
34 };
35
36 #endif //ARGENTUM_SPELL_H

```

jul 21, 20 15:47

Spell.cpp

Page 1/1

```

1 //
2 // Created by marcos on 6/13/20.
3 //
4
5 #include "Spell.h"
6 #include "../Client/GameConstants.h"
7 #include "../Miscellaneous/CameraCollisionVerifier.h"
8
9 const float ANIMATION_TIME = 20000.f;
10 const int SPELL_SPEED = 30;
11
12 Spell::Spell(Texture& texture, SDL_Rect &camera, float x, float y) :
13     sTexture(texture), camera(camera) {
14
15     currentFrame = 0;
16     timePassed = 0;
17     xPosition = x;
18     yPosition = y;
19     width = (float)TILE_WIDTH/2;
20     height = (float)TILE_HEIGHT/2 + 15;
21 }
22
23 void Spell::updateFrame(double timeStep) {
24     //Calculo time step
25     float offset = SPELL_SPEED*timeStep;
26     if ( (timePassed + offset) ≥ ANIMATION_TIME ) {
27         timePassed = ANIMATION_TIME;
28     } else {
29         timePassed += offset;
30     }
31     if (timePassed ≥ ANIMATION_TIME) {
32         currentFrame = 0;
33         timePassed = 0;
34         finished = true;
35     } else {
36         for (int i = 0; i < 24; ++i) { /*6 es la cantidad de frames distintos de
37             1 spell*/
38                 if (timePassed < ((float)ANIMATION_TIME/24 * (float)(i+1))) {
39                     currentFrame = i;
40                     break;
41                 }
42             }
43     }
44
45 void Spell::render() {
46     if (CameraCollisionVerifier::isInsideCamera(camera, {(int)xPosition,
47         (int)yPosition, (int)width, (int)height})) {
48         sTexture.render((int)(xPosition) - camera.x,
49             (int)(yPosition) - camera.y, currentFrame);
50     };
51 }
52
53 bool Spell::finishedAnimation() const {
54     return finished;
55 }
56
57 void Spell::setPosition(float x, float y) {
58     xPosition = x;
59     yPosition = y;
60 }

```

jul 21, 20 15:47

CameraCollisionVerifier.h

Page 1/1

```

1 //
2 // Created by marcos on 19/7/20.
3 //
4
5 #ifndef ARGENTUM_CAMERACOLLISIONVERIFIER_H
6 #define ARGENTUM_CAMERACOLLISIONVERIFIER_H
7
8 #include <SDL_rect.h>
9
10 class CameraCollisionVerifier {
11 public:
12     static bool isInsideCamera(SDL_Rect a, SDL_Rect b, int adjustment);
13     static bool isInsideCamera(SDL_Rect a, SDL_Rect b);
14 };
15
16
17 #endif //ARGENTUM_CAMERACOLLISIONVERIFIER_H

```

jul 21, 20 15:47

CameraCollisionVerifier.cpp

Page 1/1

```

1 //
2 // Created by marcos on 19/7/20.
3 //
4
5 #include "CameraCollisionVerifier.h"
6
7 /*Verifica si el se encuentra adentro de la camara (para renderizarlo solo
8 * si hace falta). Adjustment es para las estructuras ya que ocupan mas de un
9 * tile y las dimensiones son ligeramentes distintas*/
10
11 /*Este se usa para las estructuras*/
12 bool CameraCollisionVerifier::isInsideCamera(SDL_Rect a, SDL_Rect b, int adjustm
13 ent) {
14     int leftA, leftB;
15     int rightA, rightB;
16     int topA, topB;
17     int bottomA, bottomB;
18     //Calculo los lados de A
19     leftA = a.x;
20     rightA = a.x + a.w;
21     topA = a.y;
22     bottomA = a.y + a.h;
23
24     //Calculo los lados de B
25     leftB = b.x;
26     rightB = b.x + b.w + adjustment;
27     topB = b.y + adjustment - b.h;
28     bottomB = b.y + adjustment; /*Porque centro las estructuras en el medio del
29 tile*/
30
31     //Si alguno de los lados de A esta fuera de B
32     if(bottomA ≤ topB) return false;
33     if(topA ≥ bottomB) return false;
34     if(rightA ≤ leftB) return false;
35     if(leftA ≥ rightB) return false;
36
37     //Si ningun lado de A esta fuera de B
38     return true;
39 }
40
41 /*Este se usa para las que no son estructuras*/
42 bool CameraCollisionVerifier::isInsideCamera(SDL_Rect a, SDL_Rect b) {
43     int leftA, leftB;
44     int rightA, rightB;
45     int topA, topB;
46     int bottomA, bottomB;
47     //Calculo los lados de A
48     leftA = a.x;
49     rightA = a.x + a.w;
50     topA = a.y;
51     bottomA = a.y + a.h;
52
53     //Calculo los lados de B
54     leftB = b.x;
55     rightB = b.x + b.w;
56     topB = b.y;
57     bottomB = b.y + b.h;
58
59     if(bottomA ≤ topB) return false;
60     if(topA ≥ bottomB) return false;
61     if(rightA ≤ leftB) return false;
62     if(leftA ≥ rightB) return false;
63
64     return true;
65 }

```

jul 21, 20 15:47

Arrow.h

Page 1/1

```

1 //
2 // Created by marcos on 14/7/20.
3 //
4
5 #ifndef ARGENTUM_ARROW_H
6 #define ARGENTUM_ARROW_H
7
8 #include "../Texture/Texture.h"
9
10 /*Esta clase encapsula el comportamiento de una flecha*/
11
12 class Arrow {
13 private:
14     Texture& sTexture;
15     SDL_Rect& camera;
16     float angle, distanceToTravel;
17     float currDistance;
18     float xPosition, width;
19     float yPosition, height;
20     bool finished{false};
21
22 public:
23     Arrow(Texture& texture, SDL_Rect& camera, float x, float y,
24           float xTarget, float yTarget);
25
26     /*Renderiza la flecha (si fuera visible)*/
27     void render();
28
29     /*Actualiza la posicion de la flecha*/
30     void updatePosition(double timeStep);
31
32     /*Devuelve true si la flecha alcanzo su objetivo, asi puedo borrarla*/
33     bool reachedTarget() const;
34
35 private:
36     void _calculateTrajectory(float xTarget, float yTarget);
37 };
38
39
40 #endif //ARGENTUM_ARROW_H

```

jul 21, 20 15:47

Arrow.cpp

Page 1/1

```

1 //
2 // Created by marcos on 14/7/20.
3 //
4
5 #include "Arrow.h"
6 #include "../Client/GameConstants.h"
7 #include "../Miscellaneous/CameraCollisionVerifier.h"
8
9 const int ARROW_SPEED = 2;
10
11 Arrow::Arrow(Texture& texture, SDL_Rect &camera, float xPos, float yPos,
12             float xTarget, float yTarget) :
13     sTexture(texture), camera(camera) {
14     currDistance = 0;
15     xPosition = xPos;
16     yPosition = yPos;
17     _calculateTrajectory(xTarget, yTarget);
18     width = (float)TILE_WIDTH/2;
19     height = (float)TILE_HEIGHT/2 + 15;
20 }
21
22 void Arrow::_calculateTrajectory(float xTarget, float yTarget) {
23     float relativeXTarget = xTarget - xPosition; /*Lo llevo relativo al origen q
24 ue es la posicion de mi flecha*/
25     float relativeYTarget = yPosition - yTarget;
26     angle = atan2(relativeYTarget, relativeXTarget) * 180 / M_PI; /*Calculo el a
27 ngulo de la recta*/
28     distanceToTravel = sqrt(pow(relativeXTarget, 2.0) + pow(relativeYTarget, 2.0
29 ));
30 }
31
32 void Arrow::render() {
33     if (CameraCollisionVerifier::isInsideCamera(camera, {(int)xPosition, (int)yP
34 osition,
35             (int)width, (int)height})) {
36         sTexture.render((int)(xPosition) - camera.x,
37             (int)(yPosition) - camera.y, 0, 40 - angle);
38     }
39 }
40
41 void Arrow::updatePosition(double timeStep) {
42     if (!finished) {
43         float moved = ARROW_SPEED * timeStep;
44         xPosition += moved * cos(angle * M_PI / 180);
45         yPosition -= moved * sin(angle * M_PI / 180);
46         currDistance += moved;
47         if (currDistance >= distanceToTravel) {
48             finished = true;
49         }
50     }
51 }
52
53 bool Arrow::reachedTarget() const {
54     return finished;
55 }

```

jul 21, 20 15:47	Tile.h	Page 1/2
1	//	
2	// Created by marcos on 6/7/20.	
3	//	
4		
5	#ifndef ARGENTUM_TILE_H	
6	#define ARGENTUM_TILE_H	
7		
8	#include "../Texture/TextureRepository.h"	
9	#include "ItemDrop.h"	
10	#include "Structure.h"	
11	#include "../Character/Entity.h"	
12	#include "Coordinate.h"	
13	#include <memory>	
14	#include "../libs/GameEnums.h"	
15	#include <list>	
16	#include "../Miscellaneous/Spell.h"	
17		
18	class Tile {	
19	private:	
20	SDL_Rect box{};	
21	Texture* tileTexture{nullptr};	
22	ItemDrop item;	
23	Structure structure;	
24	std::weak_ptr<Entity> entity;	
25	std::weak_ptr<Spell> spell;	
26	int type; <i>//La mayoria de las texturas de los tiles son varias en una</i>	
27	<i>//con esto puedo especificar cual quiero en particular</i>	
28		
29	public:	
30	explicit Tile(Coordinate position);	
31		
32	<i>/*Carga la data inicial del tile, esto es, el tipo de piso y si guarda una e</i>	
33	<i>structura*/</i>	
34	void loadData(Texture& _tileTexture, Texture* sTexture = nullptr, int tileTy	
35	pe = 0);	
36		
37	<i>/*Crea un item en el tile*/</i>	
38	void createItem(Texture& _itemTexture);	
39		
40	<i>/*Renderiza el piso del tile*/</i>	
41	void renderGround(SDL_Rect& camera);	
42		
43	<i>/*Renderiza la estructura (si tuviera)*/</i>	
44	void renderStructure(SDL_Rect& camera);	
45		
46	<i>/*Renderiza la entity (si tuviera)*/</i>	
47	void renderEntity();	
48		
49	<i>/*Agrega un entity al tile*/</i>	
50	void addEntity(std::shared_ptr<Entity>& _entity);	
51		
52	<i>/*Elimina el entity del tile*/</i>	
53	void removeEntity();	
54		
55	<i>/*Agrega un spell al tile*/</i>	
56	void addSpell(std::shared_ptr<Spell>& newSpell, SDL_Rect& camera);	
57		
58	<i>/*Elimina el spell del tile*/</i>	
59	void destroyItem();	
60		
61	<i>/*Pasa el spell de entity (si tuviera) al tile. Esto es por si el entity</i>	
62	<i>* muere, para no perder la animacion*/</i>	
63	void retrieveEntitySpell();	
64	};	

jul 21, 20 15:47	Tile.h	Page 2/2
65	#endif <i>//ARGENTUM_TILE_H</i>	

jul 21, 20 15:47

Tile.cpp

Page 1/2

```

1  //
2  // Created by marcos on 6/7/20.
3  //
4
5  #include "Tile.h"
6  #include "../Client/GameConstants.h"
7
8  Tile::Tile(Coordinate position) : item(position), structure(position) {
9      box = {position.j*TILE_WIDTH, position.i*TILE_HEIGHT, TILE_WIDTH, TILE_HEIGHT};
10     type = 0;
11 }
12
13 void Tile::renderGround(SDL_Rect& camera) {
14     tileTexture->render(box.x - camera.x, box.y - camera.y, type);
15     item.render(camera);
16 }
17
18 void Tile::renderStructure(SDL_Rect &camera) {
19     structure.render(camera);
20 }
21
22 void Tile::loadData(Texture& _tileTexture, Texture* sTexture, int tileType) {
23     type = tileType;
24     tileTexture = &_tileTexture;
25     if (sTexture) {
26         structure.setTexture(*sTexture);
27     }
28 }
29
30 void Tile::createItem(Texture& _itemTexture) {
31     item.setItem(&_itemTexture);
32 }
33
34 void Tile::addEntity(std::shared_ptr<Entity>& _entity) {
35     entity = _entity;
36 }
37
38 void Tile::renderEntity() {
39     auto _entity = entity.lock();
40     if (_entity) {
41         _entity->render();
42     }
43     auto _spell = spell.lock();
44     if (_spell) {
45         _spell->render();
46     }
47 }
48
49 void Tile::removeEntity() {
50     entity.reset();
51 }
52
53 void Tile::addSpell(std::shared_ptr<Spell>& newSpell, SDL_Rect& camera) {
54     auto _entity = entity.lock();
55     if (_entity) {
56         _entity->addSpell(newSpell);
57     } else {
58         spell = newSpell;
59     }
60 }
61
62 void Tile::destroyItem() {
63     item.setItem(nullptr);
64 }
65

```

jul 21, 20 15:47

Tile.cpp

Page 2/2

```

66 void Tile::retrieveEntitySpell() {
67     auto _entity = entity.lock();
68     if (_entity) {
69         spell = _entity->getSpell();
70     }
71 }

```

jul 21, 20 15:47

Structure.h

Page 1/1

```

1  //
2  // Created by marcos on 10/6/20.
3  //
4
5  #ifndef ARGENTUM_STRUCTURE_H
6  #define ARGENTUM_STRUCTURE_H
7
8  #include "../Texture/Texture.h"
9  #include "Coordinate.h"
10
11 /*Esta clase representa una estructura en el juego, esto puede variar
12  * desde casas hasta arboles/arbustos o cadaveres*/
13
14 class Structure {
15 private:
16     SDL_Rect box{};
17     Texture* sTexture;
18
19 public:
20     explicit Structure(Coordinate position, Texture* sTexture = nullptr);
21
22     /*Setea la textura a renderizar de la estructura*/
23     void setTexture(Texture& texture);
24
25     /*Renderiza la estructura si es visible en la camara del player*/
26     void render(SDL_Rect& camera);
27 };
28
29
30 #endif //ARGENTUM_STRUCTURE_H

```

jul 21, 20 15:47

Structure.cpp

Page 1/1

```

1  //
2  // Created by marcos on 10/6/20.
3  //
4
5  #include "Structure.h"
6  #include "../Client/GameConstants.h"
7  #include "../Miscellaneous/CameraCollisionVerifier.h"
8
9  Structure::Structure(Coordinate position, Texture* sTexture) : sTexture(sTexture)
10 ) {
11     if (sTexture != nullptr) {
12         SpriteDimensions_t dimensions = sTexture->getSpriteDimensions();
13         box = {position.j*TILE_WIDTH, position.i*TILE_HEIGHT, dimensions.width,
14             dimensions.height};
15     } else {
16         box = {position.j*TILE_WIDTH, position.i*TILE_HEIGHT, 0, 0};
17     }
18
19 void Structure::render(SDL_Rect& camera) {
20     //Si se ve el tile en la pantalla
21     if (sTexture != nullptr ^ CameraCollisionVerifier::isInsideCamera(camera, box
22         , TILE_HEIGHT/2)) {
23         sTexture->render(box.x - camera.x, box.y - camera.y);
24     }
25
26 void Structure::setTexture(Texture& texture) {
27     sTexture = &texture;
28     SpriteDimensions_t dimensions = sTexture->getSpriteDimensions();
29     box.w = dimensions.width;
30     box.h = dimensions.height;
31 }

```

jul 21, 20 15:47	Map.h	Page 1/3
1	//	
2	// Created by marcos on 6/7/20.	
3	//	
4		
5	#ifndef ARGENTUM_MAP_H	
6	#define ARGENTUM_MAP_H	
7		
8	#include "Tile.h"	
9	#include <vector>	
10	#include "../Texture/TextureRepository.h"	
11	#include "Structure.h"	
12	#include "../Client/ProtocolEnumTranslator.h"	
13	#include "../Texture/PlayerEquipment.h"	
14	#include <list>	
15	#include "../Client/EntityData.h"	
16	#include "../Miscellaneous/Spell.h"	
17	#include "../Client/CitizenData.h"	
18	#include "../Sound/SoundPlayer.h"	
19	#include "../Miscellaneous/Arrow.h"	
20		
21	/*Esta clase representa al mapa del juego y tiene el ownership de las entidades,	
22	* flechas y hechizos en juego*/	
23		
24	class Map {	
25	private:	
26	TextureRepository& textureRepo;	
27	SoundPlayer& soundPlayer;	
28	std::vector<Tile> tiles;	
29	SDL_Rect& camera;	
30	std::unordered_map<std::string, std::pair<std::shared_ptr<Entity>, Coordinat	
31	e>> entities;	
32	std::list<std::tuple<std::shared_ptr<Entity>*, Coordinate, std::string>> ent	
33	itiesToUpdateTilePosition;	
34	/*Esto es para no pisar entities entre si cuando terminan de moverse*/	
35		
36	std::list<std::shared_ptr<Spell>> spells;	
37	std::list<std::unique_ptr<Arrow>> arrows;	
38	std::string playerName;	
39	public:	
40	Map(TextureRepository& repo, SDL_Rect& camera, SoundPlayer& soundPlayer);	
41		
42	/*Setea el tamaño del vector unidimensional tiles*/	
43	void setSize(int rows, int columns);	
44		
45	/*Carga la data inicial del tile position, es decir, su tipo de piso,	
46	* estructura (si tuviera) y citizen (si tuviera). Es lo primero que recibo	
47	* cuando me conecto al server, para cada tile*/	
48	void loadTileData(Coordinate position, FloorTypeTexture floor, TextureID str	
49	ucture,	
50	CitizenData& citizen);	
51		
52	/*Crea el item con la textura pedida en el tile de coordenada position*/	
53	void createItem(Coordinate position, TextureID itemTexture);	
54		
55	/*Crea un npc en el mapa en base a la data provista (textura, posicion, etc)	
56	*/	
57	void addNPC(EntityData& data);	
58		
59	/*Crea un player en el mapa en base a la data provista (textura, posicion, e	
60	tc)*/	
	void addPlayer(MapPlayerData& playerData);	
	/*Desplaza a la entity en la direccion recibida la distanceTravelled indicad	
	a,	
	* , si termino de moverse (reachedDestination = true) le resetea la animaci	

jul 21, 20 15:47	Map.h	Page 2/3
61	on.	
62	* Si es el entity estaba quieto entonces lo encola a la	
63	* lista de entitesToUpdateTilePosition para actualizarle su posicion*/	
64	void moveEntity(std::string& nickname, GameType::Direction direction,	
65	unsigned int distanceTravelled, bool reachedDestination);	
66		
67	/*Centra la camara en la entidad cuyo nombre matchee con nickname	
68	* (se utiliza para el propio player)*/	
69	void setCameraOn(std::string& nickname);	
70		
71	/*Borra del mapa a la entity*/	
72	void removeEntity(std::string& nickname);	
73		
74	/*Equipa en el EquipmentPlace indicado el equipment recibido, se usa para	
75	* cambiar lo que tienen equipados los players)*/	
76	void equipOnPlayer(std::string& nickname, GameType::EquipmentPlace place,	
77	TextureID equipment);	
78		
79	/*Setea la texture del player en el fantasma*/	
80	void killPlayer(std::string& nickname);	
81		
82	/*Setea la texture del player en la que tiene cuando esta vivo, es decir,	
83	* si tiene la del fantasma se la quita*/	
84	void revivePlayer(std::string& nickname);	
85		
86	/*Agrego un spell en la posicion indicada. Si hubiera un player en dicha	
87	* posicion se lo agrega a el, sino lo agrega al tile*/	
88	void addSpell(Coordinate position, TextureID spellTexture);	
89		
90	/*Agrega un arrow a la lista de arrows, seteando su posicion inicial en	
91	* la del arquero, dicha arrow se movera al target a medida que pase el tiem	
92	po*/	
93	void addArrow(std::string& archerNickname, Coordinate target, TextureID arro	
94	wTexture);	
95		
96	/*Actualiza las animaciones del juego en base al timeStep y mueve a los enti	
97	ties	
98	* a sus nuevos tiles correspondientes*/	
99	void update(double timeStep = 0);	
100		
101	/*Elimina el item en la posicion indicada*/	
102	void destroyItem(Coordinate itemPosition);	
103		
104	/*Teletransporta un entity de una posicion a otra. Se usa para cuando	
105	* en el juego un player revive en una ciudad*/	
106	void teleportEntity(const std::string& nickname, Coordinate newPosition,	
107	bool isMyPlayer);	
108		
109	/*Setea el nickname del player propio, util para poder referenciarlo y saber	
110	* que sonidos realizar (lo buscamos en el map y conseguimos su ubicacion	
111	* en el mapa)*/	
112	void setPlayerNickname(const std::string& nickname);	
113		
114	/*Verifica si el sonido a realizar esta dentro del rango de sonido del playe	
115	r	
116	* propio*/	
117	void verifyQueueSound(Coordinate tile, SoundID sound, int maxDistance);	
118		
119	/*Cambia la direccion de renderizado del entity, se utiliza en el evento de	
	un ataque	
	* para que mire a su target*/	
	void changeEntityLookDirection(std::string& nickname, GameType::Direction di	
	rection);	
	/*Actualiza el nivel que se muestra del player*/	
	void updatePlayerLevel(const std::string& _playerNickname, int level);	

jul 21, 20 15:47

Map.h

Page 3/3

```

120
121     /*Renderiza el mapa, esto incluye entities, estructuras, spells, etc*/
122     void render();
123
124 private:
125     static Coordinate _calculateNewTile(Coordinate position, GameType::Direction
direction);
126     void _updateSpells(double timeStep);
127     void _updateArrows(double timeStep);
128     void _moveEntitiesToNewTile();
129     void _addEntity(EntityData& data, std::shared_ptr<Entity>& entity);
130     void _renderGround();
131     void _renderStructures();
132     void _renderNPCS();
133     void _renderArrows();
134 };
135
136
137 #endif //ARGENTUM_MAP_H

```

jul 21, 20 15:47

Map.cpp

Page 1/6

```

1  //
2  // Created by marcos on 6/7/20.
3  //
4
5  #include "Map.h"
6  #include <fstream>
7  #include <algorithm>
8  #include "../Client/GameConstants.h"
9  #include "../Character/NPC.h"
10 #include "../Character/Player.h"
11
12 Map::Map(TextureRepository& repo, SDL_Rect& camera, SoundPlayer& soundPlayer) :
13     textureRepo(repo), soundPlayer(soundPlayer), camera(camera) {
14     this->camera = camera;
15 }
16
17 void Map::_renderGround() {
18     for (int i = 0; i < (VISIBLE_VERTICAL_TILES + 1); ++i) {
19         for (int j = 0; j < (VISIBLE_HORIZONTAL_TILES + 1); ++j) {
20             float x = (float)camera.x + (float)j * TILE_WIDTH;
21             float y = (float)camera.y + (float)i * TILE_HEIGHT;
22             if (x ≥ LEVEL_WIDTH ∨ x < 0) continue;
23             if (y ≥ LEVEL_HEIGHT ∨ y < 0) continue;
24             int xTile = floor(x / TILE_WIDTH);
25             int yTile = floor(y / TILE_HEIGHT);
26             int tile = yTile * TOTAL_HORIZONTAL_TILES + xTile;
27             tiles.at(tile).renderGround(camera);
28         }
29     }
30 }
31
32 void Map::_renderStructures() {
33     for (int i = -1; i < (VISIBLE_VERTICAL_TILES + 3); ++i) {
34         for (int j = -1; j < (VISIBLE_HORIZONTAL_TILES + 3); ++j) {
35             float x = (float)camera.x + (float)j * TILE_WIDTH;
36             float y = (float)camera.y + (float)i * TILE_HEIGHT;
37             if (x ≥ LEVEL_WIDTH ∨ x < 0) continue;
38             if (y ≥ LEVEL_HEIGHT ∨ y < 0) continue;
39             int xTile = floor(x / TILE_WIDTH);
40             int yTile = floor(y / TILE_HEIGHT);
41             int tile = yTile * TOTAL_HORIZONTAL_TILES + xTile;
42             tiles.at(tile).renderStructure(camera);
43         }
44     }
45 }
46
47 void Map::_renderNPCS() {
48     for (int i = -2; i < (VISIBLE_VERTICAL_TILES + 2); ++i) {
49         for (int j = -2; j < (VISIBLE_HORIZONTAL_TILES + 2); ++j) {
50             float x = (float)camera.x + (float)j * TILE_WIDTH;
51             float y = (float)camera.y + (float)i * TILE_HEIGHT;
52             if (x ≥ LEVEL_WIDTH ∨ x < 0) continue;
53             if (y ≥ LEVEL_HEIGHT ∨ y < 0) continue;
54             int xTile = floor(x / TILE_WIDTH);
55             int yTile = floor(y / TILE_HEIGHT);
56             int tile = yTile * TOTAL_HORIZONTAL_TILES + xTile;
57             tiles.at(tile).renderEntity();
58         }
59     }
60 }
61
62 void Map::setSize(int rows, int columns) {
63     for (int i = 0; i < rows; ++i) {
64         for (int j = 0; j < columns; ++j) {
65             tiles.emplace_back(Coordinate{i, j});
66         }
67     }
68 }

```


jul 21, 20 15:47

Map.cpp

Page 2/6

```

67     }
68 }
69
70 void Map::loadTileData(Coordinate position, FloorTypeTexture floor, TextureID st
ructure,
71     CitizenData& citizen) {
72     unsigned int tile = position.i*TOTAL_HORIZONTAL_TILES + position.j;
73     if (structure != Nothing) {
74         tiles.at(tile).loadData(textureRepo.getTexture(floor.texture),
75             &textureRepo.getTexture(structure), floor.index);
76     } else {
77         tiles.at(tile).loadData(textureRepo.getTexture(floor.texture),
78             nullptr, floor.index);
79     }
80     if (citizen.texture != Nothing) {
81         EntityData data = {citizen.texture, std::move(citizen.nickname), positio
n,
82             GameType::DIRECTION_STILL, 0};
83         addNPC(data);
84     }
85 }
86
87 void Map::_addEntity(EntityData& data, std::shared_ptr<Entity>& entity) {
88     if (data.currentDir != GameType::DIRECTION_STILL) {
89         Coordinate destination = _calculateNewTile(data.pos,
90             data.currentDir);
91         entity->move(data.currentDir, data.distanceMoved, false);
92         entities.emplace(data.nickname, std::make_pair(std::move(entity),
93             data.pos));
94         entitiesToUpdateTilePosition.emplace_back(&entities.at(data.nickname).fi
rst,
95             destination, data.nickname);
96     } else {
97         int tile = data.pos.i*TOTAL_HORIZONTAL_TILES + data.pos.j;
98         tiles.at(tile).addEntity(entity);
99         entities.emplace(data.nickname, std::make_pair(std::move(entity),
100             data.pos));
101     }
102 }
103
104 void Map::addNPC(EntityData& data) {
105     if (entities.count(data.nickname) == 0) {
106         std::string npcLevel;
107         if (data.level > 0) {
108             npcLevel = std::to_string(data.level);
109         }
110         std::shared_ptr<Entity> npc = std::make_shared<NPC>(textureRepo,
111             camera, data.pos.j*TILE_WIDTH, data.pos.i*TILE_HEIGHT, data.textu
re,
112             std::move(npcLevel));
113         _addEntity(data, npc);
114     }
115 }
116
117 void Map::addPlayer(MapPlayerData& playerData) {
118     if (entities.count(playerData.entityData.nickname) == 0) {
119         std::shared_ptr<Entity> player = std::make_shared<Player>(textureRepo, ca
mera,
120             playerData.entityData.pos.j*TILE_WIDTH, playerData.entityData.pos
.i*TILE_HEIGHT,
121             playerData.equipment, playerData.isAlive, std::to_string(playerD
ata.entityData.level),
122             playerData.entityData.nickname);
123         _addEntity(playerData.entityData, player);
124     }
125 }

```

jul 21, 20 15:47

Map.cpp

Page 3/6

```

126
127 void Map::createItem(Coordinate position, TextureID itemTexture) {
128     unsigned int tile = position.i*TOTAL_HORIZONTAL_TILES + position.j;
129     tiles.at(tile).createItem(textureRepo.getTexture(itemTexture));
130 }
131
132 void Map::moveEntity(std::string &nickname, GameType::Direction direction,
133     unsigned int distanceTravelled, bool reachedDestination) {
134
135     if (entities.count(nickname) == 1) { /*Si no lo mataron en el update*/
136         Entity* entity = entities.at(nickname).first.get();
137         GameType::Direction previousDirection = entity->move(direction,
138             distanceTravelled, reachedDestination);
139         if (previousDirection == GameType::DIRECTION_STILL) { /*Se empezo a mover
de tile*/
140             Coordinate oldPosition = entities.at(nickname).second;
141             verifyQueueSound(oldPosition, StepSound, 3);
142             Coordinate newPosition = _calculateNewTile(oldPosition, direction);
143             int tile = oldPosition.i*TOTAL_HORIZONTAL_TILES + oldPosition.j;
144             tiles.at(tile).removeEntity();
145             entitiesToUpdateTilePosition.emplace_back(&entities.at(nickname).fir
st,
146                 newPosition, std::move(nickn
ame));
147         }
148     }
149 }
150
151 void Map::verifyQueueSound(Coordinate tile, SoundID sound, int maxDistance) {
152     Coordinate playerPos = entities.at(playerNickname).second;
153     int distance = std::abs(playerPos.j - tile.j) + std::abs(playerPos.i - tile.
i);
154     if (distance <= maxDistance) {
155         soundPlayer.queueSound(sound);
156     }
157 }
158
159 Coordinate Map::_calculateNewTile(Coordinate position, GameType::Direction direc
tion) {
160     switch (direction) {
161         case GameType::DIRECTION_UP:
162             position.i -= 1;
163             break;
164         case GameType::DIRECTION_DOWN:
165             position.i += 1;
166             break;
167         case GameType::DIRECTION_LEFT:
168             position.j -= 1;
169             break;
170         case GameType::DIRECTION_RIGHT:
171             position.j += 1;
172             break;
173         case GameType::DIRECTION_STILL:
174             //do nothing
175             break;
176     }
177     return position;
178 }
179
180 void Map::_moveEntitiesToNewTile() {
181     if (!entitiesToUpdateTilePosition.empty()) {
182         for (auto & entity : entitiesToUpdateTilePosition) {
183             if (entities.count(std::get<2>(entity)) == 1) { //este chequeo es par
a el posible caso donde lo mueven al
184                 // nuevo til
e y matan en el mismo update

```

jul 21, 20 15:47

Map.cpp

Page 4/6

```

185     int tile = std::get<1>(entity).i * TOTAL_HORIZONTAL_TILES + std::
: get<1>(entity).j;
186     entities.at(std::get<2>(entity)).second = std::get<1>(entity);
187     tiles.at(tile).addEntity(*std::get<0>(entity));
188 }
189 }
190 entitiesToUpdateTilePosition.clear();
191 }
192 }
193
194 void Map::setCameraOn(std::string& nickname) {
195     entities.at(nickname).first->activateCamera();
196 }
197
198 void Map::removeEntity(std::string &nickname) {
199     Coordinate position = entities.at(nickname).second;
200     int tile = position.i * TOTAL_HORIZONTAL_TILES + position.j;
201     tiles.at(tile).retrieveEntitySpell();
202     tiles.at(tile).removeEntity();
203     entities.erase(nickname);
204 }
205
206 void Map::equipOnPlayer(std::string &nickname, GameType::EquipmentPlace place,
207     TextureID equipment) {
208     Entity* entity = entities.at(nickname).first.get();
209     auto player = dynamic_cast<Player*>(entity);
210     if (player) {
211         player->equip(place, equipment);
212     }
213 }
214
215 void Map::killPlayer(std::string &nickname) {
216     Entity* entity = entities.at(nickname).first.get();
217     Coordinate position = entities.at(nickname).second;
218     verifyQueueSound(position, Death1Sound, 10);
219     auto player = dynamic_cast<Player*>(entity);
220     if (player) {
221         player->kill();
222     }
223 }
224
225 void Map::revivePlayer(std::string &nickname) {
226     Entity* entity = entities.at(nickname).first.get();
227     auto player = dynamic_cast<Player*>(entity);
228     if (player) {
229         player->revive();
230     }
231 }
232
233 void Map::addSpell(Coordinate position, TextureID spellTexture) {
234     int tile = position.i * TOTAL_HORIZONTAL_TILES + position.j;
235     std::shared_ptr<Spell> spell(new Spell(textureRepo.getTexture(spellTexture),
236         camera, position.j * TILE_WIDTH, position.i * TILE_HEIGHT)); //No uso mak
e shared ya que el spell
237 //no
lo borro del weak ptr al destruirse
238 // y
con make shared eso resultaria en conservar mas memoria
239 tiles.at(tile).addSpell(spell, camera);
240 spells.emplace_back(std::move(spell));
241 }
242
243 void Map::addArrow(std::string& archerNickname, Coordinate target, TextureID arr
owTexture) {
244     Coordinate archerPosition = entities.at(archerNickname).second;
245     arrows.emplace_back(new Arrow(textureRepo.getTexture(arrowTexture), camera,

```

jul 21, 20 15:47

Map.cpp

Page 5/6

```

246     archerPosition.j * TILE_WIDTH, archerPosition.i * TILE_HEIGHT,
247     target.j * TILE_WIDTH, target.i * TILE_HEIGHT));
248 }
249
250 void Map::update(double timeStep) {
251     _moveEntitiesToNewTile();
252     _updateSpells(timeStep);
253     _updateArrows(timeStep);
254 }
255
256 static bool shouldSpellBeRemoved(std::shared_ptr<Spell>& spell) {
257     if (spell) {
258         return spell->finishedAnimation();
259     }
260     return true;
261 }
262
263 static bool shouldArrowBeRemoved(std::unique_ptr<Arrow>& arrow) {
264     if (arrow) {
265         return arrow->reachedTarget();
266     }
267     return true;
268 }
269
270 void Map::updateArrows(double timeStep) {
271     if (!arrows.empty()) {
272         for (auto & arrow : arrows) {
273             if (arrow) {
274                 arrow->updatePosition(timeStep);
275             }
276             arrows.erase(std::remove_if(arrows.begin(), arrows.end(),
277                 shouldArrowBeRemoved), arrows.end());
278         }
279     }
280 }
281
282 void Map::updateSpells(double timeStep) {
283     if (!spells.empty()) {
284         for (auto & spell : spells) {
285             if (spell) {
286                 spell->updateFrame(timeStep);
287             }
288             spells.erase(std::remove_if(spells.begin(), spells.end(),
289                 shouldSpellBeRemoved), spells.end());
290         }
291     }
292 }
293
294 void Map::destroyItem(Coordinate itemPosition) {
295     int tile = itemPosition.i * TOTAL_HORIZONTAL_TILES + itemPosition.j;
296     tiles.at(tile).destroyItem();
297 }
298
299 void Map::teleportEntity(const std::string &nickname, Coordinate newPosition,
300     bool isMyPlayer) {
301     Entity* entity = entities.at(nickname).first.get();
302     Coordinate oldPosition = entities.at(nickname).second;
303     int oldTile = oldPosition.i * TOTAL_HORIZONTAL_TILES + oldPosition.j;
304     tiles.at(oldTile).removeEntity();
305     int newTile = newPosition.i * TOTAL_HORIZONTAL_TILES + newPosition.j;
306     entity->setPosition(newPosition.j * TILE_WIDTH, newPosition.i * TILE_HEIGHT);
307     if (isMyPlayer) {
308         entity->activateCamera();
309     }
310     tiles.at(newTile).addEntity(entities.at(nickname).first);
311     entities.at(nickname).second = newPosition;

```

jul 21, 20 15:47

Map.cpp

Page 6/6

```

312 }
313
314 void Map::setPlayerNickname(const std::string &nickname) {
315     playerNickname = nickname;
316 }
317
318 void Map::changeEntityLookDirection(std::string& nickname, GameType::Direction d
319     irection) {
320     Entity* entity = entities.at(nickname).first.get();
321     entity->setLookDirection(direction);
322 }
323
324 void Map::render() {
325     _renderGround();
326     _renderNPCS();
327     _renderArrows();
328     _renderStructures();
329 }
330
331 void Map::_renderArrows() {
332     for (auto & arrow : arrows) {
333         arrow->render();
334     }
335 }
336
337 void Map::updatePlayerLevel(const std::string& _playerNickname, int level) {
338     Entity* player = entities.at(_playerNickname).first.get();
339     player->updateLevel(level);
340 }

```

jul 21, 20 15:47

ItemDrop.h

Page 1/1

```

1  //
2  // Created by marcos on 6/13/20.
3  //
4
5  #ifndef ARGENTUM_ITEMDROP_H
6  #define ARGENTUM_ITEMDROP_H
7
8  #include "../Texture/Texture.h"
9  #include "Coordinate.h"
10
11  /*Esta clase representa la imagen particular de un item (el drop). Se renderiza
12   * en el mapa cuando esta droppeado en un tile o tambien en el inventario*/
13
14  class ItemDrop {
15  private:
16      SDL_Rect box{};
17      Texture* sTexture;
18
19  public:
20      explicit ItemDrop(Coordinate position);
21
22      /*Renderiza el item*/
23      void render(SDL_Rect& camera);
24
25      /*Setea la textura del item a renderizar*/
26      void setItem(Texture* itemTexture);
27  };
28
29
30 #endif //ARGENTUM_ITEMDROP_H

```

jul 21, 20 15:47

ItemDrop.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/13/20.
3  //
4
5  #include "ItemDrop.h"
6  #include "../Client/GameConstants.h"
7
8  ItemDrop::ItemDrop(Coordinate position) : sTexture(nullptr) {
9      box = {position.j*TILE_WIDTH, position.i*TILE_HEIGHT, 0, 0};
10 }
11
12 void ItemDrop::render(SDL_Rect& camera) {
13     if (sTexture != nullptr) {
14         sTexture->render(box.x - camera.x, box.y - camera.y, 0, -90);
15     }
16 }
17
18 void ItemDrop::setItem(Texture* itemTexture) {
19     if (itemTexture != nullptr) {
20         SpriteDimensions_t dimensions = itemTexture->getSpriteDimensions();
21         box.w = dimensions.width;
22         box.h = dimensions.height;
23     }
24     sTexture = itemTexture;
25 }

```

jul 21, 20 15:47

Coordinate.h

Page 1/1

```

1  //
2  // Created by marcos on 6/27/20.
3  //
4
5  #ifndef ARGENTUM_COORDINATE_H
6  #define ARGENTUM_COORDINATE_H
7
8  struct Coordinate {
9      int i;
10     int j;
11
12     bool operator!=(Coordinate& other) {
13         return (i != other.i || j != other.j);
14     }
15 };
16
17 #endif //ARGENTUM_COORDINATE_H

```

jul 21, 20 15:47	Text.h	Page 1/1
1	//	
2	// Created by ivan on 12/6/20.	
3	//	
4		
5	#include "Font.h"	
6	#include "../Texture/Texture.h"	
7		
8	#include <string>	
9		
10	#ifndef ARGENTUM_TEXT_H	
11	#define ARGENTUM_TEXT_H	
12		
13		
14	class Text {	
15	private:	
16	Font& font;	
17	std::string text;	
18	Texture textTexture;	
19		
20	public:	
21	Text(Font& font, SDL_Renderer& renderer, std::string& _text = "");	
22		
23	Text(Font& font, SDL_Renderer& renderer, const std::string& _text);	
24		
25	Text(Text& other) noexcept;	
26		
27	/* Setea el texto a "newText" */	
28	Text& updateText(std::string& newText);	
29		
30	/* Setea el texto a "newText" */	
31	Text& updateText(const std::string& newText);	
32		
33	/* Agrega "newText" al final del texto */	
34	Text& operator+=(std::string& newText);	
35		
36	/* Borra la ultima letra del texto */	
37	Text& operator--();	
38		
39	/* Me devuelve el tamaño del texto */	
40	int getLength();	
41		
42	/* Me devuelve una referencia al texto */	
43	std::string& getText();	
44		
45	/* Renderiza el texto en la posicion (x,y) con el color "color" */	
46	void render(int x, int y);	
47		
48	/* Genera la textura en base al texto almacenado */	
49	Text& operator*(SDL_Color color);	
50		
51	/* Es para mandarle un color default al otro, C++ no me deja ponerle default sino */	
52	Text& operator*();	
53		
54	/*Retorna el ancho en pixeles del texto, se usa para dejar lindo el nombre c	
55	on el nivel*/	
56	int getTextTextureWidth();	
57		
58	~Text();	
59	};	
60		
61	#endif //ARGENTUM_TEXT_H	

jul 21, 20 15:47	Text.cpp	Page 1/2
1	//	
2	// Created by ivan on 12/6/20.	
3	//	
4		
5	#include "Text.h"	
6		
7	Text::Text(Font& font, SDL_Renderer& renderer, std::string& _text) : font(font)	
8	{	
9	text = std::move(_text);	textTexture(renderer) {
10	operator*();	
11	}	
12		
13	Text& Text::updateText(std::string& newText) {	
14	text = std::move(newText);	
15	return *this;	
16	}	
17		
18	Text& Text::operator+=(std::string& newText) {	
19	text += newText;	
20	return *this;	
21	}	
22		
23	Text& Text::operator*(SDL_Color color) {	
24	if (!text.empty()) {	
25	textTexture.loadFromRenderedText(text, color, font.getFont());	
26	}	
27	return *this;	
28	}	
29		
30	void Text::render(int x, int y) {	
31	if (!text.empty()) {	
32	textTexture.render(x, y);	
33	}	
34	}	
35		
36	Text& Text::operator--() {	
37	if (!text.empty()) {	
38	text.pop_back();	
39	}	
40	return *this;	
41	}	
42		
43	int Text::getLength() {	
44	return text.length();	
45	}	
46		
47	std::string &Text::getText() {	
48	return text;	
49	}	
50		
51	Text& Text::operator*() {	
52	return operator*({0xFF, 0xFF, 0xFF});	
53	}	
54		
55	Text &Text::updateText(const std::string &newText) {	
56	text = newText;	
57	return *this;	
58	}	
59		
60	Text::Text(Font& font, SDL_Renderer& renderer, const std::string& _text) : font(font),	
61		textTexture
62	{	
63	text = _text;	
	operator*();	

jul 21, 20 15:47

Text.cpp

Page 2/2

```

64 }
65
66 int Text::getTextTextureWidth() {
67     if (!text.empty()) {
68         return textTexture.getSpriteDimensions().width;
69     } else {
70         return 0;
71     }
72 }
73
74 Text::Text(Text &other) noexcept : font(other.font), text(std::move(other.text)
75 ),
76                                     textTexture(std::move(other.textTexture)) {}
77
78 Text::~Text() = default;

```

jul 21, 20 15:47

Font.h

Page 1/1

```

1  #ifndef ARGENTUM_FONT_H
2  #define ARGENTUM_FONT_H
3
4  #include <SDL.h>
5  #include <SDL_image.h>
6  #include <SDL_ttf.h>
7  #include <string>
8  #include <iostream>
9
10 #include "../libs/TPEException.h"
11
12 class Font {
13 private:
14     TTF_Font *font;
15 public:
16     Font(const std::string& path, int fontSize);
17
18     TTF_Font* getFont();
19
20     ~Font();
21 };
22
23
24 #endif // ARGENTUM_FONT_H

```

jul 21, 20 15:47

Font.cpp

Page 1/1

```

1  //
2  // Created by ivan on 7/6/20.
3  //
4
5  #include "Font.h"
6
7  Font::Font(const std::string& path, int fontSize) {
8      //carga la font
9      font = TTF_OpenFont(path.c_str(), fontSize);
10     if(font == nullptr){
11         throw TPException( "Failed to load lazy font! SDL_ttf Error: %s\n",
12                             TTF_GetError() );
13     }
14 }
15
16 TTF_Font *Font::getFont() {
17     return this->font;
18 }
19
20 Font::~Font() {
21     TTF_CloseFont(font);
22 }

```

jul 21, 20 15:47

Selector.h

Page 1/1

```

1  //
2  // Created by ivan on 20/6/20.
3  //
4
5  #ifndef ARGENTUM_SELECTOR_H
6  #define ARGENTUM_SELECTOR_H
7
8  #include <SDL.h>
9  #include "../Screen/Window.h"
10 #include "../Map/Coordinate.h"
11 #include "../libs/GameEnums.h"
12 #include <mutex>
13
14 //Maneja las cosas que selecciona el usuario
15 class Selector {
16 private:
17     //Coordinate clickX;
18     Coordinate inventorySlot;
19     Coordinate selectedTile;
20     GameType::EquipmentPlace selectedEquipment;
21     std::mutex m;
22
23 public:
24     Selector();
25
26     //Se fija si el click fue en el mapa o en el inventario. Dependiendo el caso
27     //Guarda el tile/inventorySlot/equipable que se haya seleccionado.
28     void handleEvent(Coordinate click, Coordinate playerPos, Window& window);
29
30     //Devuelve el inventorySlot actualmente seleccionado
31     int getInventorySlot();
32
33     //Devuelve la coordenada del tile actualmente seleccionado
34     Coordinate getSelectedTile();
35
36     //Me devuelve el item equipado que seleccione
37     GameType::EquipmentPlace getSelectedEquipment();
38
39     //Devuelve true si seleccione un tile
40     static bool hasSelectedTile(Coordinate click);
41
42     //Devuelve true si seleccione un slot del inventario
43     static bool hasSelectedSlot(Coordinate click);
44
45     //Devuelve true si seleccione un item equipado
46     static bool hasSelectedEquipment(Coordinate click);
47
48     //Setea el tile seleccionado a (0,0)
49     void resetTileSelection();
50
51     ~Selector();
52
53 private:
54     void _verifyTileSelection(Coordinate playerPos, Coordinate click);
55     void _verifyInventorySlotSelection(Coordinate click);
56     static bool _isInsideRect(Coordinate click, int left, int right, int top,
57                               int bottom) ;
58     void _verifySelectedEquipment(Coordinate click);
59
60 };
61
62
63
64 #endif //ARGENTUM_SELECTOR_H

```

jul 21, 20 15:47

Selector.cpp

Page 1/3

```

1  //
2  // Created by ivan on 20/6/20.
3  //
4
5  #include "../Client/GameConstants.h"
6  #include "Selector.h"
7  #include <iostream>
8
9  #define DEFAULT_MAP_LEFT 20
10 #define DEFAULT_MAP_RIGHT 1044
11 #define DEFAULT_MAP_TOP 236
12 #define DEFAULT_MAP_BOTTOM 876
13
14 #define CAMERA_X_OFFSET 27
15 #define CAMERA_Y_OFFSET 15
16
17 #define DEFAULT_INVENTORY_LEFT 1122
18 #define DEFAULT_INVENTORY_RIGHT 1434
19 #define DEFAULT_INVENTORY_TOP 260
20 #define DEFAULT_INVENTORY_BOTTOM 548
21
22 #define INVENTORY_SLOT_WIDTH 78
23 #define INVENTORY_SLOT_HEIGHT 72
24
25
26 Selector::Selector() {
27     inventorySlot = {0, 0};
28     selectedTile = {0, 0};
29 }
30
31
32 void Selector::handleEvent(Coordinate click, Coordinate playerPos, Window& window) {
33     std::lock_guard<std::mutex> l(m);
34     _verifyTileSelection(playerPos, click);
35     _verifyInventorySlotSelection(click);
36     _verifySelectedEquipment(click);
37 }
38
39 void Selector::_verifyTileSelection(Coordinate playerPos, Coordinate click) {
40     //Veo si clickeo adentro del mapa
41     if (_isInsideRect(click, DEFAULT_MAP_LEFT, DEFAULT_MAP_RIGHT, DEFAULT_MAP_TOP,
42         DEFAULT_MAP_BOTTOM)){
43         //Esto es cuando no esta en los extremos
44         int playerXTile = playerPos.j;
45         int playerYTile = playerPos.i;
46         int relativeXTile = (click.j - DEFAULT_MAP_LEFT + CAMERA_X_OFFSET) / TILE_WIDTH;
47         int relativeYTile = (click.i - DEFAULT_MAP_TOP - CAMERA_Y_OFFSET) / TILE_HEIGHT;
48         selectedTile.j = playerXTile + (relativeXTile - 4);
49         selectedTile.i = playerYTile + (relativeYTile - 2);
50
51         //Me fijo los extremos
52         if (playerXTile < 4){
53             selectedTile.j = (click.j - DEFAULT_MAP_LEFT) / TILE_WIDTH;
54         } else if (playerXTile > 95){
55             selectedTile.j = 92 + ((click.j - DEFAULT_MAP_LEFT) / TILE_WIDTH);
56         }
57         if (playerYTile < 3){
58             selectedTile.i = (click.i - DEFAULT_MAP_TOP) / TILE_HEIGHT;
59         } else if (playerYTile > 97){
60             selectedTile.i = 95 + ((click.i - DEFAULT_MAP_TOP) / TILE_HEIGHT);
61         }
62     }
63 }

```

jul 21, 20 15:47

Selector.cpp

Page 2/3

```

63 }
64
65 void Selector::_verifyInventorySlotSelection(Coordinate click) {
66     //Veo si clickeo adentro del inventario
67     if (_isInsideRect(click, DEFAULT_INVENTORY_LEFT, DEFAULT_INVENTORY_RIGHT,
68         DEFAULT_INVENTORY_TOP, DEFAULT_INVENTORY_BOTTOM)){
69         inventorySlot.j = (click.j - DEFAULT_INVENTORY_LEFT) / INVENTORY_SLOT_WIDTH;
70         inventorySlot.i = (click.i - DEFAULT_INVENTORY_TOP) / INVENTORY_SLOT_HEIGHT;
71     }
72 }
73
74 void Selector::_verifySelectedEquipment(Coordinate click) {
75     if (_isInsideRect(click, 1320, 1392, 660, 735)){//Casco
76         selectedEquipment = GameType::EQUIPMENT_PLACE_HEAD;
77     } else if (_isInsideRect(click, 1397, 1469, 660, 735)){//Chest
78         selectedEquipment = GameType::EQUIPMENT_PLACE_CHEST;
79     } else if (_isInsideRect(click, 1320, 1392, 760, 835)){//Weapon
80         selectedEquipment = GameType::EQUIPMENT_PLACE_WEAPON;
81     } else if (_isInsideRect(click, 1397, 1469, 760, 835)){//Shield
82         selectedEquipment = GameType::EQUIPMENT_PLACE_SHIELD;
83     }
84 }
85
86 bool Selector::hasSelectedTile(Coordinate click) {
87     return _isInsideRect(click, DEFAULT_MAP_LEFT, DEFAULT_MAP_RIGHT, DEFAULT_MAP_TOP,
88         DEFAULT_MAP_BOTTOM);
89 }
90
91 bool Selector::hasSelectedSlot(Coordinate click) {
92     return _isInsideRect(click, DEFAULT_INVENTORY_LEFT, DEFAULT_INVENTORY_RIGHT,
93         DEFAULT_INVENTORY_TOP, DEFAULT_INVENTORY_BOTTOM);
94 }
95
96 bool Selector::hasSelectedEquipment(Coordinate click) {
97     //Aca me fijo si esta entre los 4 cuadrados de equipamiento
98     return _isInsideRect(click, 1320, 1469,
99         660, 835);
100 }
101
102 int Selector::getInventorySlot() {
103     std::lock_guard<std::mutex> l(m);
104     return inventorySlot.j + (4 * inventorySlot.i);
105 }
106
107 Coordinate Selector::getSelectedTile() {
108     std::lock_guard<std::mutex> l(m);
109     return {selectedTile.i, selectedTile.j};
110 }
111
112 GameType::EquipmentPlace Selector::getSelectedEquipment() {
113     std::lock_guard<std::mutex> l(m);
114     return selectedEquipment;
115 }
116
117 void Selector::resetTileSelection() {
118     std::lock_guard<std::mutex> l(m);
119     selectedTile = {0, 0};
120 }
121
122 bool Selector::_isInsideRect(Coordinate click, int left, int right, int top, int bottom) {
123     return (click.j > left ^ click.j < right ^ click.i > top ^ click.i < bottom);
124 }

```


jul 21, 20 15:47

Selector.cpp

Page 3/3

```

124 }
125
126
127 Selector::~Selector() = default;

```

jul 21, 20 15:47

Minichat.h

Page 1/2

```

1  //
2  // Created by ivan on 18/6/20.
3  //
4
5  #ifndef ARGENTUM_MINICHAT_H
6  #define ARGENTUM_MINICHAT_H
7
8  #include "../Text/Text.h"
9  #include "../Screen/Window.h"
10 #include "../Client/GameConstants.h"
11 #include <list>
12 #include "../Map/Coordinate.h"
13 #include <mutex>
14 #include <atomic>
15
16 class Minichat {
17 private:
18     Font minichatFont;
19     Text input;
20     std::atomic<bool> processedInput{false};
21     std::list<Text> texts;
22     SDL_Renderer& renderer;
23     bool focusOnMinichat;
24     int firstToRender;
25     std::mutex generalMutex;
26     std::mutex inputMutex;
27
28 public:
29     explicit Minichat(SDL_Renderer& renderer);
30
31     //Encola newText para imprimirlo en el minichat
32     void queueText(std::string &newText);
33
34     //Recibe texto y si es necesario lo separa en varias lineas del minichat
35     void receiveText(const std::string &text);
36
37     //Renderiza todos los mensajes del minichat + lo que escribio el usuario
38     void render();
39
40     //Borra la ultima letra del input
41     void handleBackspace();
42
43     //Asigna el texto del evento de input al texto de input del usuario
44     void handleTextInput(SDL_Event &e);
45
46     //Se fija si el click fue dentro del minichat. Si es asi habilita el input d
47     e texto
48     void handleMouseButtonDown(Coordinate click, Window &window);
49
50     //Scrollea por los mensajes del minichat
51     void handleMouseWheel(SDL_Event &e);
52
53     //Borra los mensajes del minichat
54     void clearMinichat();
55
56     /* Devuelve el input del usuario en el minichat para que sea parseado como c
57     omando */
58     std::string handleReturnKey();
59
60     ~Minichat();
61
62 private:
63     static bool _isInsideMinichat(int xClick, int yClick);
64     void _queueInputIfProcessed();
65 };

```

jul 21, 20 15:47

Minichat.h

Page 2/2

```
65 #endif //ARGENTUM_MINICHAT_H
```

jul 21, 20 15:47

Minichat.cpp

Page 1/3

```
1 //
2 // Created by ivan on 18/6/20.
3 //
4
5 #include "Minichat.h"
6
7 #define MINICHAT_X_OFFSET 15
8 #define MINICHAT_Y_OFFSET 15
9
10 #define MAX_TEXT_LEN 85
11 #define MAX_MSGS 24 //El maximo de mensajes que se van a ver al scrollear
12 #define MAX_MSGS_TO_RENDER 8
13
14 #define MINICHAT_FONT_PATH "/var/Argentum/Assets/Fonts/Raleway-Medium.ttf"
15
16 Minichat::Minichat(SDL_Renderer& renderer) : minichatFont(MINICHAT_FONT_PATH, 20
17 ),
18                                     input(minichatFont,renderer), renderer(renderer)
19 {
20     focusOnMinichat = false;
21     input.updateText(":");
22
23     //Lleno el vector con mensajes vacios
24     for (int i = 0; i < MAX_MSGS; ++i) {
25         texts.emplace_back(minichatFont,renderer);
26     }
27     firstToRender = 0;
28
29 std::string Minichat::handleReturnKey() {
30     std::lock_guard<std::mutex> l(inputMutex);
31     std::string cmd = input.getText();
32     if (cmd.size() > 1) {
33         cmd.erase(0, 1); //Le saco ":"
34         input.getText().erase(0, 1);
35         processedInput = true;
36         return cmd;
37     }
38     return "";
39 }
40
41 void Minichat::handleBackspace() {
42     std::lock_guard<std::mutex> l(inputMutex);
43     if (focusOnMinichat) {
44         if (input.getTextLength() > 1) {
45             --input;
46         }
47     }
48 }
49
50 void Minichat::handleTextInput(SDL_Event &e) {
51     std::lock_guard<std::mutex> l(inputMutex);
52     std::string newInput = e.text.text;
53     if (input.getTextLength() < MAX_TEXT_LEN) {
54         input += std::move(newInput);
55     }
56 }
57
58 void Minichat::handleMouseButtonDown(Coordinate click, Window& window) {
59     focusOnMinichat = _isInsideMinichat(click.j, click.i);
60     if (focusOnMinichat) {
61         SDL_StartTextInput();
62     } else {
63         SDL_StopTextInput();
64     }
65 }
```

jul 21, 20 15:47

Minichat.cpp

Page 2/3

```

65
66 void Minichat::handleMouseWheel(SDL_Event& e) {
67     std::lock_guard<std::mutex> l(generalMutex);
68     if (focusOnMinichat) {
69         if (e.wheel.y > 0) // scroll up
70         {
71             firstToRender += 1;
72             if (firstToRender > MAX_MSGS - MAX_MSGS_TO_RENDER)
73                 firstToRender -= 1;
74         } else if (e.wheel.y < 0) // scroll down
75         {
76             firstToRender -= 1;
77             if (firstToRender < 0)
78                 firstToRender = 0;
79         }
80     }
81 }
82
83 void Minichat::receiveText(const std::string& text) {
84     //Separa un mensaje separado con \n en varias lineas
85     int currPos = -1;
86     int nextPos;
87     while (currPos < (int)text.size()) {
88         nextPos = text.find('\n', currPos + 1);
89         std::string substr = text.substr(currPos + 1, nextPos - currPos - 1);
90         currPos = nextPos;
91         queueText(substr);
92         if (nextPos == -1)
93             break;
94     }
95 }
96
97 //Imprime los mensajes relevantes
98 void Minichat::queueText(std::string& newText) {
99     std::lock_guard<std::mutex> l(generalMutex);
100     if (!newText.empty()) {
101         texts.pop_back();
102         texts.emplace_front(minichatFont, renderer);
103         *(texts.front().updateText(std::move(newText)));
104     }
105 }
106
107 void Minichat::clearMinichat() {
108     std::lock_guard<std::mutex> l(generalMutex);
109     for (auto & text : texts) {
110         text.updateText("");
111     }
112 }
113
114 void Minichat::_queueInputIfProcessed() {
115     if (processedInput) { /*No lo encola el thread de input porque SDL no puede
116         actualizar texturas desde otro thread*/
117         queueText(input.getText());
118         input.updateText("");
119         processedInput = false;
120     }
121 }
122
123 void Minichat::render() {
124     std::lock_guard<std::mutex> li(inputMutex);
125     _queueInputIfProcessed();
126     std::lock_guard<std::mutex> lg(generalMutex);
127     (*input).render(0,178);
128     //renderizo mensajes encolados.
129     int textNum = 0;
130     for (auto & text : texts) {

```

jul 21, 20 15:47

Minichat.cpp

Page 3/3

```

130         if (textNum >= firstToRender) {
131             if (!text.getText().empty()) {
132                 text.render(0,140 - 20*(textNum - firstToRender));
133             }
134         }
135         ++textNum;
136         if (textNum >= MAX_MSGS_TO_RENDER + firstToRender) break;
137     }
138 }
139
140 bool Minichat::_isInsideMinichat(int xClick, int yClick){
141     return (xClick >= MINICHAT_X_OFFSET) ^
142            (xClick < (MINICHAT_X_OFFSET + DEFAULT_MINICHAT_WIDTH))
143            ^ ( yClick >= MINICHAT_Y_OFFSET )
144            ^ ( yClick <= (MINICHAT_Y_OFFSET + DEFAULT_MINICHAT_HEIGHT));
145 }
146
147 Minichat::~Minichat() = default;
148
149

```

jul 21, 20 15:47

PlayerStats.h

Page 1/1

```

1 //
2 // Created by marcos on 7/2/20.
3 //
4
5 #ifndef ARGENTUM_PLAYERSTATS_H
6 #define ARGENTUM_PLAYERSTATS_H
7
8 #include "../Map/Coordinate.h"
9
10 struct PlayerStats {
11     std::string nickname;
12     int32_t totalHealth, totalMana, nextLevelXP;
13     int32_t health, mana, xp;
14     int32_t level;
15     int32_t constitution, strength, agility, intelligence;
16     int32_t gold, safeGold;
17     Coordinate position;
18 };
19 enum EquippedItems {
20     Helmet,
21     Armor,
22     Weapon,
23     Shield,
24 };
25
26 #endif //ARGENTUM_PLAYERSTATS_H

```

jul 21, 20 15:47

PlayerInventoryGUI.h

Page 1/1

```

1 //
2 // Created by ivan on 13/6/20.
3 //
4
5 #ifndef ARGENTUM_PLAYERINVENTORYGUI_H
6 #define ARGENTUM_PLAYERINVENTORYGUI_H
7
8 #include "../Texture/TextureRepository.h"
9 #include "../Text/Text.h"
10 #include "PlayerInfoGUI.h"
11 #include <list>
12 #include <unordered_map>
13
14 struct FixedText {
15     Text inventory;
16     Text title;
17
18     FixedText(SDL_Renderer& renderer, Font& font) : inventory(font, renderer, "I
NVENTORY"),
19                                                     title(font, renderer, "Argentum")
20 };
21
22 class PlayerInventoryGUI {
23 private:
24     Font textFont;
25     Text text;
26     TextureRepository& repo;
27     SDL_Renderer& renderer;
28     std::vector<Texture*> inventoryTextures;
29     std::unordered_map<EquippedItems, Texture*> equippedTextures;
30     PlayerInfoGUI& pInfo;
31     FixedText fixedText;
32     int32_t gold, safeGold{};
33
34 public:
35     PlayerInventoryGUI(TextureRepository& repo, SDL_Renderer& renderer, PlayerIn
foGUI& pInfo);
36
37     /* Agrega la textura del item al slot correspondiente */
38     void addInventoryItem(TextureID texture, int32_t slot);
39
40     /* Agrega un item a la posicion correspondiente de los equipables */
41     void addEquipableItem(TextureID texture, EquippedItems item);
42
43     /* Actualiza la cantidad de oro y oro seguro */
44     void updateGold(int32_t gold, int32_t _safeGold);
45
46     /* Renderiza el inventario y la informacion del jugador */
47     void render(int32_t selectedSlot);
48
49 private:
50     void _drawInventoryOutlines(int32_t x);
51     void _renderInventoryItems();
52     void _drawEquipableOutlines();
53     void _renderEquipableItems();
54     void _renderText();
55     void _renderSkills();
56 };
57
58
59 #endif //ARGENTUM_PLAYERINVENTORYGUI_H

```

jul 21, 20 15:47

PlayerInventoryGUI.cpp

Page 1/3

```

1  //
2  // Created by ivan on 13/6/20.
3  //
4
5  #include "PlayerInventoryGUI.h"
6
7  #define INVENTORY_SIZE 16
8
9  #define INVENTORY_ITEMS_X_OFFSET 45
10 #define INVENTORY_ITEMS_Y_OFFSET 235
11
12 #define ITEM_WIDTH 72
13 #define ITEM_HEIGHT 75
14
15 #define INVENTORY_OUTLINES_X_OFFSET 73
16 #define INVENTORY_OUTLINES_Y_OFFSET 260
17
18 #define INVENTORY_FONT_PATH "/var/Argentum/Assets/Fonts/medieval.ttf"
19
20 PlayerInventoryGUI::PlayerInventoryGUI(TextureRepository &repo,SDL_Renderer &renderer,
21                                     PlayerInfoGUI& playerInfo) :
22     textFont(INVENTORY_FONT_PATH, 25),
23     text(textFont,renderer), repo(repo),
24     renderer(renderer), pInfo(playerInfo)
25 {
26     gold = 0;
27     for (int i = 0; i < INVENTORY_SIZE; ++i) { //Inicializo el vector con nullptr
28         inventoryTextures.push_back(nullptr);
29     }
30     equippedTextures[Helmet] = nullptr;
31     equippedTextures[Armor] = nullptr;
32     equippedTextures[Weapon] = nullptr;
33     equippedTextures[Shield] = nullptr;
34 }
35
36 void PlayerInventoryGUI::addInventoryItem(TextureID texture, int32_t slot) {
37     if (texture == Nothing) {
38         inventoryTextures[slot] = nullptr;
39     } else {
40         inventoryTextures[slot] = &repo.getTexture(texture);
41     }
42 }
43
44 void PlayerInventoryGUI::addEquipableItem(TextureID texture, EquipableItems item)
45 {
46     if (texture != Nothing) {
47         Texture* currTexture = &repo.getTexture(texture);
48         equippedTextures.at(item) = currTexture;
49     } else {
50         equippedTextures.at(item) = nullptr;
51     }
52 }
53
54 void PlayerInventoryGUI::updateGold(int32_t _gold, int32_t _safeGold) {
55     gold = _gold;
56     safeGold = _safeGold;
57 }
58
59 void PlayerInventoryGUI::render(int32_t selectedSlotX) {
60     _drawInventoryOutlines(selectedSlotX);
61     _drawEquipableOutlines();
62     _renderInventoryItems();
63     _renderEquipableItems();
64     _renderText();

```

jul 21, 20 15:47

PlayerInventoryGUI.cpp

Page 2/3

```

64 }
65
66 void PlayerInventoryGUI::_renderText() {
67     fixedText.title.render(215, 25);
68     fixedText.inventory.render(160, 225);
69     pInfo.getGoldText().render(140, 565);
70     pInfo.getLevelText().render(70, 50);
71     _renderSkills();
72     pInfo.getPositionText().render(200, 880);
73     pInfo.getNicknameText().render(210,95);
74 }
75
76 void PlayerInventoryGUI::_renderSkills() {
77     pInfo.getStrengthText().render(40, 660);
78     pInfo.getConstitutionText().render(40, 700);
79     pInfo.getIntelligenceText().render(40, 740);
80     pInfo.getAgilityText().render(40, 780);
81 }
82
83 void PlayerInventoryGUI::_renderInventoryItems() {
84     for (int i = 0; i < 4; i++) {
85         for (int j = 0; j < 4; ++j) {
86             if (inventoryTextures[4*i + j] != nullptr){
87                 inventoryTextures[4*i + j]→render(INVENTORY_ITEMS_X_OFFSET +
88                                                     (ITEM_WIDTH + 6) * j,INVENTORY_ITEMS_Y_O
89                                                     FFSET
90                                                     + (i) * (ITEM_HEIGHT - 1),0);
91             }
92         }
93     }
94 }
95
96 void PlayerInventoryGUI::_renderEquipableItems() {
97     if (equippedTextures.at(Helmet)) {
98         equippedTextures.at(Helmet)→render(250, 635,0);
99     }
100     if (equippedTextures.at(Armor)) {
101         equippedTextures.at(Armor)→render(325, 635, 0);
102     }
103     if (equippedTextures.at(Weapon)) {
104         equippedTextures.at(Weapon)→render(250, 735, 0);
105     }
106     if (equippedTextures.at(Shield)) {
107         equippedTextures.at(Shield)→render(325, 735, 0);
108     }
109 }
110
111 /* Renderiza los cuadrados en los que se divide el inventario */
112 void PlayerInventoryGUI::_drawInventoryOutlines(int32_t selectedSlot) {
113     SDL_Rect outlineRect;
114     for (int i = 0; i < 4; ++i) {
115         for (int j = 0; j < 4; ++j) {
116             outlineRect = { INVENTORY_OUTLINES_X_OFFSET + (ITEM_WIDTH + 6) * j,
117                             INVENTORY_OUTLINES_Y_OFFSET + (ITEM_HEIGHT - 3) * i,
118                             ITEM_WIDTH + 6, ITEM_HEIGHT - 3 };
119             SDL_SetRenderDrawColor(&renderer, 0x3f, 0x2a,
120                                     0x14, 0xFF);
121             if ((j + 4 * i) == selectedSlot){
122                 SDL_SetRenderDrawColor(&renderer, 0xff, 0xff,
123                                         0xff, 0xFF);
124             }
125             SDL_RenderDrawRect( &renderer, &outlineRect );
126         }
127     }
128 }

```

jul 21, 20 15:47

PlayerInventoryGUI.cpp

Page 3/3

```

129  /* Renderiza los cuadrados donde se muestran los items equipados */
130  void PlayerInventoryGUI::_drawEquipableOutlines() {
131      SDL_Rect outlineRect;
132      for (int j = 0; j < 2; ++j) {
133          outlineRect = { 275 + 75 * j, 660, ITEM_WIDTH, ITEM_HEIGHT };
134          SDL_SetRenderDrawColor( &renderer, 0x3f,0x2a,
135                                  0x14, 0xFF );
136          SDL_RenderDrawRect( &renderer, &outlineRect );
137      }
138      for (int j = 0; j < 2; ++j) {
139          outlineRect = { 275 + 75 * j, 760, ITEM_WIDTH, ITEM_HEIGHT };
140          SDL_SetRenderDrawColor( &renderer, 0x3f,0x2a,
141                                  0x14, 0xFF );
142          SDL_RenderDrawRect( &renderer, &outlineRect );
143      }
144  }
145  }

```

jul 21, 20 15:47

PlayerInfoGUI.h

Page 1/2

```

1  //
2  // Created by ivan on 13/6/20.
3  //
4
5  #ifndef ARGENTUM_PLAYERINFOGUI_H
6  #define ARGENTUM_PLAYERINFOGUI_H
7
8  #include "../Text/Text.h"
9  #include "PlayerStats.h"
10 #include "../Map/Coordinate.h"
11 #include "../Sound/SoundPlayer.h"
12
13 struct GUIInfoText {
14     Text nickname;
15     Text health;
16     Text mana;
17     Text xp;
18     Text level;
19     Text constitution, strength, agility, intelligence;
20     Text gold;
21     Text position;
22
23     GUIInfoText(SDL_Renderer& renderer, Font& font) : nickname(font, renderer),
24     health(font, renderer), mana(font, renderer),
25     xp(font, renderer), level(font, renderer),
26     constitution(font, renderer),
27     strength(font, renderer), agility(font, renderer),
28     intelligence(font, renderer), gold(font, renderer),
29     position(font, renderer) {}
30 };
31
32 class PlayerInfoGUI {
33 private:
34     Font infoFont;
35     Text info;
36     SDL_Renderer& renderer;
37     PlayerStats pInfo{};
38     SoundPlayer& soundPlayer;
39     GUIInfoText infoText;
40
41 public:
42     PlayerInfoGUI(SDL_Renderer& renderer, SoundPlayer& soundPlayer);
43
44     Text& getLevelText();
45     int32_t getXPos() const;
46     int32_t getYPos() const;
47     std::string& getNickname();
48     Text& getStrengthText();
49     Text& getConstitutionText();
50     Text& getAgilityText();
51     Text& getIntelligenceText();
52     Text& getPositionText();
53     Text& getNicknameText();
54     Text& getGoldText();
55
56     /* Actualiza todas las stats del jugador */
57     void update(PlayerStats& generalInfo);
58
59     /* Renderiza las barras de vida, xp y mana */
60     void render();
61
62 private:
63     void _renderInfoBar(Text& textToRender, int32_t infoCurr, int32_t infoTotal,
64         int32_t xOffset,
65         int32_t barLen, SDL_Color color);
66     void _updateHealth(int32_t currHealth, int32_t totalHealth);

```

jul 21, 20 15:47

PlayerInfoGUI.h

Page 2/2

```

66 void _updateMana(int32_t currMana, int32_t totalMana);
67 void _updateXP(int32_t currXP, int32_t nextLevelXP);
68 void _updateLevel(int32_t newLevel);
69 void _updatePosition(Coordinate position);
70 void _updateStrength(int32_t strength);
71 void _updateConstitution(int32_t constitution);
72 void _updateAgility(int32_t agility);
73 void _updateIntelligence(int32_t intelligence);
74 void _updateNickname(std::string^ name);
75 void _updateGold(int32_t gold, int32_t safeGold);
76 };
77
78
79 #endif //ARGENTUM_PLAYERINFOGUI_H

```

jul 21, 20 15:47

PlayerInfoGUI.cpp

Page 1/4

```

1  //
2  // Created by ivan on 13/6/20.
3  //
4
5  #include "PlayerInfoGUI.h"
6
7  #include <utility>
8  #include "../Client/GameConstants.h"
9
10 #define PLAYER_INFO_FONT_PATH "/var/Argentum/Assets/Fonts/medieval.ttf"
11 #define HEALTH_TEXT "HEALTH: " + std::to_string(pInfo.health) + "/" + std::to_string(pInfo.totalHealth)
12 #define MANA_TEXT "MANA: " + std::to_string(pInfo.mana) + "/" + std::to_string(pInfo.totalMana)
13 #define XP_TEXT "XP: " + std::to_string(pInfo.xp) + "/" + std::to_string(pInfo.nextLevelXP)
14 #define POSITION_TEXT "X: " + std::to_string(getXPos()) + " " + "Y: " + std::to_string(getYPos())
15 #define GOLD_TEXT "GOLD: " + std::to_string(pInfo.gold) + "(" + std::to_string(pInfo.safeGold) + ")"
16
17 PlayerInfoGUI::PlayerInfoGUI(SDL_Renderer &renderer, SoundPlayer& soundPlayer) : infoFont(PLAYER_INFO_FONT_PATH, 25), info(infoFont, renderer), renderer(renderer), soundPlayer(soundPlayer), infoText(renderer, infoFont) {
18
19     pInfo = {};
20     *(infoText.health.updateText(HEALTH_TEXT));
21     *(infoText.mana.updateText(MANA_TEXT));
22     *(infoText.xp.updateText(XP_TEXT));
23 }
24
25 void PlayerInfoGUI::_updateHealth(int32_t currHealth, int32_t totalHealth) {
26     if (pInfo.health != currHealth || pInfo.totalHealth != totalHealth) {
27         pInfo.health = currHealth;
28         pInfo.totalHealth = totalHealth;
29         *(infoText.health.updateText(HEALTH_TEXT));
30     }
31 }
32
33 void PlayerInfoGUI::_updateMana(int32_t currMana, int32_t totalMana) {
34     if (pInfo.mana != currMana || pInfo.totalMana != totalMana) {
35         pInfo.mana = currMana;
36         pInfo.totalMana = totalMana;
37         *(infoText.mana.updateText(MANA_TEXT));
38     }
39 }
40
41 void PlayerInfoGUI::_updateXP(int32_t currXP, int32_t nextLevelXP) {
42     if (pInfo.xp != currXP || pInfo.nextLevelXP != nextLevelXP) {
43         pInfo.xp = currXP;
44         pInfo.nextLevelXP = nextLevelXP;
45         *(infoText.xp.updateText(XP_TEXT));
46     }
47 }
48
49 void PlayerInfoGUI::_updateLevel(int32_t newLevel) {
50     if (pInfo.level != newLevel) {
51         if (newLevel > pInfo.level) {
52             soundPlayer.queueSound(LevelUpSound);
53         }
54         pInfo.level = newLevel;
55         *(infoText.level.updateText(std::to_string(pInfo.level)));
56     }
57 }
58
59
60

```

jul 21, 20 15:47

PlayerInfoGUI.cpp

Page 2/4

```

61 void PlayerInfoGUI::updateStrength(int32_t strength) {
62     if (pInfo.strength != strength) {
63         pInfo.strength = strength;
64         *(infoText.strength.updateText("STRENGTH: "
65                                         + std::to_string(pInfo.strength)));
66     }
67 }
68
69 void PlayerInfoGUI::updateAgility(int32_t agility) {
70     if (pInfo.agility != agility) {
71         pInfo.agility = agility;
72         *(infoText.agility.updateText("AGILITY: "
73                                       + std::to_string(pInfo.agility)));
74     }
75 }
76
77 void PlayerInfoGUI::updateConstitution(int32_t constitution) {
78     if (pInfo.constitution != constitution) {
79         pInfo.constitution = constitution;
80         *(infoText.constitution.updateText("CONSTITUTION: "
81                                            + std::to_string(pInfo.constitution)));
82     }
83 }
84
85 void PlayerInfoGUI::updateIntelligence(int32_t intelligence) {
86     if (pInfo.intelligence != intelligence) {
87         pInfo.intelligence = intelligence;
88         *(infoText.intelligence.updateText("INTELLIGENCE: "
89                                            + std::to_string(pInfo.intelligence))
90     );
91 }
92
93 void PlayerInfoGUI::updatePosition(Coordinate position) {
94     if (pInfo.position != position) {
95         pInfo.position = position;
96         *(infoText.position.updateText(POSITION_TEXT));
97     }
98 }
99
100 void PlayerInfoGUI::updateNickname(std::string^ name) {
101     if (pInfo.nickname != name) {
102         pInfo.nickname = std::move(name);
103         *(infoText.nickname.updateText(pInfo.nickname));
104     }
105 }
106
107 void PlayerInfoGUI::updateGold(int32_t gold, int32_t safeGold) {
108     if (pInfo.gold != gold || pInfo.safeGold != safeGold) {
109         pInfo.gold = gold;
110         pInfo.safeGold = safeGold;
111         (infoText.gold.updateText(GOLD_TEXT)).operator*(
112             SDL_Color{0xFF,0xFF,0x00});
113     }
114 }
115
116 void PlayerInfoGUI::render() {
117     _renderInfoBar(infoText.health, pInfo.health, pInfo.totalHealth,
118                   HEALTH_BAR_X_OFFSET, 265,{0x99, 0x00,0x00});
119
120     _renderInfoBar(infoText.mana,
121                   pInfo.mana, pInfo.totalMana, MANA_BAR_X_OFFSET, 265,{0x00, 0x33, 0x6
122     6});
123
124     _renderInfoBar(infoText.xp,
125                   pInfo.xp, pInfo.nextLevelXP, XP_BAR_X_OFFSET, 265,{0x00, 0x66, 0x00}

```

jul 21, 20 15:47

PlayerInfoGUI.cpp

Page 3/4

```

125     };
126 }
127 void PlayerInfoGUI::_renderInfoBar(Text& textToRender, int32_t infoCurr, int32_t
128 infoTotal,
129                                     int32_t xOffset, int32_t barLen, SDL_Color co
130 lor) {
131     float bar = 0;
132     if (infoTotal != 0){
133         bar = barLen * ((float)infoCurr / (float)infoTotal);
134     }
135     //Barra
136     SDL_Rect fillRect = {xOffset, 10, (int)bar, BAR_HEIGHT};
137     SDL_SetRenderDrawColor(&renderer, color.r, color.g, color.b, 0xFF);
138     SDL_RenderFillRect(&renderer, &fillRect);
139
140     //outline de la barra
141     SDL_Rect outlineRect = {xOffset, 10, (int)barLen, BAR_HEIGHT};
142     SDL_SetRenderDrawColor(&renderer, 0x00,0x00,0x00, 0xFF );
143     SDL_RenderDrawRect(&renderer, &outlineRect );
144     //Texto de la barra
145     textToRender.render(xOffset, 10);
146 }
147 Text& PlayerInfoGUI::getLevelText() {
148     return infoText.level;
149 }
150
151 int32_t PlayerInfoGUI::getXPos() const {
152     return pInfo.position.j;
153 }
154
155 int32_t PlayerInfoGUI::getYPos() const {
156     return pInfo.position.i;
157 }
158
159 Text& PlayerInfoGUI::getStrengthText() {
160     return infoText.strength;
161 }
162
163 Text& PlayerInfoGUI::getConstitutionText() {
164     return infoText.constitution;
165 }
166
167 Text& PlayerInfoGUI::getAgilityText() {
168     return infoText.agility;
169 }
170
171 Text& PlayerInfoGUI::getIntelligenceText() {
172     return infoText.intelligence;
173 }
174
175 Text& PlayerInfoGUI::getNicknameText() {
176     return infoText.nickname;
177 }
178
179 std::string& PlayerInfoGUI::getNickname() {
180     return pInfo.nickname;
181 }
182
183 void PlayerInfoGUI::update(PlayerStats &generalInfo) {
184     _updateHealth(generalInfo.health, generalInfo.totalHealth);
185     _updateMana(generalInfo.mana, generalInfo.totalMana);
186     _updateXP(generalInfo.xp, generalInfo.nextLevelXP);
187     _updateLevel(generalInfo.level);

```


jul 21, 20 15:47

PlayerInfoGUI.cpp

Page 4/4

```

188     _updatePosition(generalInfo.position);
189     _updateStrength(generalInfo.strength);
190     _updateConstitution(generalInfo.constitution);
191     _updateAgility(generalInfo.agility);
192     _updateIntelligence(generalInfo.intelligence);
193     _updateGold(generalInfo.gold, generalInfo.safeGold);
194     _updateNickname(std::move(generalInfo.nickname));
195 }
196
197 Text &PlayerInfoGUI::getGoldText() {
198     return infoText.gold;
199 }
200
201 Text &PlayerInfoGUI::getPositionText() {
202     return infoText.position;
203 }

```

jul 21, 20 15:47

UpdateReceiver.h

Page 1/1

```

1  //
2  // Created by marcos on 6/29/20.
3  //
4
5  #ifndef ARGENTUM_UPDATERECEIVER_H
6  #define ARGENTUM_UPDATERECEIVER_H
7
8  #include "../libs/Thread.h"
9  #include "Update.h"
10 #include "ClientProtocol.h"
11 #include <msgpack.hpp>
12
13 class UpdateEvent;
14 class Socket;
15 class UpdateManager;
16
17 class UpdateReceiver : public Thread {
18 private:
19     ClientProtocol& protocol;
20     UpdateManager& updateManager;
21     Update currentUpdate;
22     msgpack::object_handle handler;
23     std::size_t offset{0};
24     Socket& socket;
25     std::vector<char> buffer;
26     bool& quit;
27
28 public:
29     UpdateReceiver(ClientProtocol& protocol, UpdateManager& _updateManager,
30                   Socket& _socket, bool& _quit) : protocol(protocol),
31                                                  updateManager(_updateManager), socket(_socket), quit(_quit) {}
32
33     void run() override;
34
35 private:
36     void _processAttack();
37     void _processCreateItem();
38     void _processUnequip();
39     void _processUpdate(uint32_t msgLength);
40     void _processRemoveEntity();
41     void _processMoveUpdate();
42     void _receivePlayerData();
43     void _processCreateEntity();
44     void _processEquipped();
45     void _processPlayerDeath();
46     void _processDestroyItem();
47     void _processTeleportEntity();
48     void _processPlayerResurrect();
49     void _processPlayerLevelUp();
50 };
51
52
53 #endif // ARGENTUM_UPDATERECEIVER_H

```

jul 21, 20 15:47

UpdateReceiver.cpp

Page 1/4

```

1  //
2  // Created by marcos on 6/29/20.
3  //
4  #include <netinet/in.h>
5  #include "UpdateReceiver.h"
6  #include "../libs/Socket.h"
7  #include "../UpdateEvents/UpdateMove.h"
8  #include "../UpdateEvents/UpdateCreatePlayer.h"
9  #include "../UpdateEvents/UpdateCreateNPC.h"
10 #include "../UpdateEvents/UpdateGUI.h"
11 #include "../UpdateEvents/UpdateRemoveEntity.h"
12 #include "../UpdateEvents/UpdateEquip.h"
13 #include "../UpdateEvents/UpdateCreateItem.h"
14 #include "../UpdateEvents/UpdatePlayerDeath.h"
15 #include "../UpdateEvents/UpdateAttack.h"
16 #include "../UpdateEvents/UpdateDestroyItem.h"
17 #include "../UpdateEvents/UpdateTeleportEntity.h"
18 #include "../UpdateEvents/UpdatePlayerResurrect.h"
19 #include "../UpdateEvents/UpdateLevelUp.h"
20 #include "UpdateManager.h"
21
22 MSGPACK_ADD_ENUM(GameType::EventID)
23 MSGPACK_ADD_ENUM(GameType::Direction)
24 MSGPACK_ADD_ENUM(GameType::Entity)
25 MSGPACK_ADD_ENUM(GameType::EquipmentPlace)
26 MSGPACK_ADD_ENUM(GameType::ItemType)
27 MSGPACK_ADD_ENUM(GameType::Weapon)
28
29 /* Recibe un update del server, lo procesa y lo encola en una queue de functors
30 * para que sea ejecutado en el thread principal */
31 void UpdateReceiver::run() {
32     try {
33         uint32_t msgLength = 0;
34         while (!quit) {
35             offset = 0;
36             socket.receive((char *) (&msgLength), sizeof(uint32_t));
37             msgLength = ntohs(msgLength);
38             buffer.clear();
39             buffer.resize(msgLength);
40             socket.receive(buffer.data(), msgLength);
41             _processUpdate(msgLength);
42             updateManager.push(currentUpdate);
43         }
44     } catch (std::exception& e) {
45         std::cerr << e.what() << std::endl;
46     } catch (...) {
47         std::cerr << "Unkown error in UpdateReceiver" << std::endl;
48     }
49     quit = true;
50 }
51
52 /* Chequea que tipo de evento recibio y lo procesa. Luego recibe toda la informacion del jugador */
53 void UpdateReceiver::_processUpdate(uint32_t msgLength) {
54     msgpack::type::tuple<GameType::EventID> id;
55     while (offset < msgLength) {
56         handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
57         handler->convert(id);
58         switch (std::get<0>(id)) {
59             case GameType::MOVED:
60                 _processMoveUpdate();
61                 break;
62             case GameType::ATTACK:
63                 _processAttack();
64                 break;
65             case GameType::UNEQUIP:

```

jul 21, 20 15:47

UpdateReceiver.cpp

Page 2/4

```

66         _processUnequip();
67         break;
68     case GameType::EQUIPPED:
69         _processEquipped();
70         break;
71     case GameType::CREATE_ENTITY:
72         _processCreateEntity();
73         break;
74     case GameType::CREATE_ITEM:
75         _processCreateItem();
76         break;
77     case GameType::REMOVE_ENTITY:
78         _processRemoveEntity();
79         break;
80     case GameType::PLAYER_DEATH:
81         _processPlayerDeath();
82         break;
83     case GameType::DESTROY_ITEM:
84         _processDestroyItem();
85         break;
86     case GameType::TELEPORTED:
87         _processTeleportEntity();
88         break;
89     case GameType::RESURRECTED:
90         _processPlayerResurrect();
91         break;
92     case GameType::PLAYER_LEVEL_UP:
93         _processPlayerLevelUp();
94         break;
95     default:
96         std::cerr << std::get<0>(id) << " is an unknown command" << std::endl;
97         break;
98     }
99 }
100 _receivePlayerData();
101 }
102
103 void UpdateReceiver::_processPlayerLevelUp() {
104     msgpack::type::tuple<std::string, int32_t> playerData;
105     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
106     handler->convert(playerData);
107     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateLevelUp(
108         std::move(std::get<0>(playerData)),
109         std::get<1>(playerData))));
110 }
111
112 void UpdateReceiver::_processTeleportEntity() {
113     msgpack::type::tuple<std::string, int32_t, int32_t> teleportData;
114     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
115     handler->convert(teleportData);
116     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateTeleportEntity(
117         std::move(std::get<0>(teleportData)),
118         {std::get<1>(teleportData),
119          std::get<2>(teleportData)})));
120 }
121
122 void UpdateReceiver::_processDestroyItem() {
123     msgpack::type::tuple<int32_t, int32_t> itemPosition;
124     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
125     handler->convert(itemPosition);
126     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateDestroyItem({std::
127         get<0>(itemPosition),
128         std::get<1>(itemPosition)})));
129 }

```

jul 21, 20 15:47

UpdateReceiver.cpp

Page 3/4

```

130 void UpdateReceiver::_processAttack() {
131     msgpack::type::tuple<std::string, int32_t, int32_t, GameType::Weapon,
132         GameType::Direction> entity;
133     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
134     handler->convert(entity);
135     currentUpdate.push(std::unique_ptr<UpdateEvent>(
136         new UpdateAttack(std::get<0>(entity),
137             {std::get<1>(entity), std::get<2>(entity)},
138             std::get<3>(entity), std::get<4>(entity))));
139 }
140
141
142 void UpdateReceiver::_processPlayerDeath() {
143     msgpack::type::tuple<std::string> player;
144     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
145     handler->convert(player);
146     currentUpdate.push(std::unique_ptr<UpdateEvent>(
147         new UpdatePlayerDeath(std::move(std::get<0>(player)))));
148 }
149
150 void UpdateReceiver::_processPlayerResurrect() {
151     msgpack::type::tuple<std::string> player;
152     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
153     handler->convert(player);
154     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdatePlayerResurrect(
155         std::move(std::get<0>(player)))));
156 }
157
158 void UpdateReceiver::_processCreateItem() {
159     msgpack::type::tuple<GameType::ItemType, int32_t, int32_t, int32_t> itemData
160     ;
161     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
162     handler->convert(itemData);
163     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateCreateItem(std::ge
164         t<0>(itemData),
165             std::get<1>(itemData), {std::get<2>(itemData), std::get<3>(itemD
166             ata)})));
167 }
168
169 void UpdateReceiver::_processUnequip() {
170     msgpack::type::tuple<std::string, GameType::EquipmentPlace> data;
171     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
172     handler->convert(data);
173     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateEquip(std::move(st
174         d::get<0>(data)),
175             std::get<1>(data), UNEQUIP)));
176 }
177
178 void UpdateReceiver::_processEquipped() {
179     msgpack::type::tuple<std::string, GameType::EquipmentPlace, int32_t> data;
180     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
181     handler->convert(data);
182     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateEquip(std::move(st
183         d::get<0>(data),
184             std::get<1>(data), std::get<2>(data))));
185 }
186
187 void UpdateReceiver::_processMoveUpdate() {
188     msgpack::type::tuple<GameType::Direction, int32_t, std::string, bool> moveIn
189     fo;
190     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
191     handler->convert(moveInfo);
192     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateMove(std::move(std
193         ::get<2>(moveInfo)),
194             std::get<0>(moveInfo), std::get<1>(moveInfo), std::get<3>(moveInfo))
195     ));

```

jul 21, 20 15:47

UpdateReceiver.cpp

Page 4/4

```

188 }
189
190 void UpdateReceiver::_processRemoveEntity() {
191     msgpack::type::tuple<std::string> nickname;
192     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
193     handler->convert(nickname);
194     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateRemoveEntity(
195         std::move(std::get<0>(nickname))));
196 }
197
198 void UpdateReceiver::_processCreateEntity() {
199     handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
200     msgpack::type::tuple<GameType::Entity, std::string, int32_t> entityData;
201     handler->convert(entityData);
202     if (std::get<0>(entityData) != GameType::PLAYER) {
203         EntityData data = protocol.processAddNPC(&buffer, entityData, offset);
204         currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateCreateNPC(data
205             ));
206     } else {
207         MapPlayerData data = protocol.processAddPlayer(&buffer, entityData, offs
208             et);
209         currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateCreatePlayer(d
210             ata));
211     }
212 }
213
214 /* Recibe la informacion del jugador para poder mostrarla en la interfaz grafica
215 */
216 void UpdateReceiver::_receivePlayerData() {
217     uint32_t length = 0;
218     socket.receive(reinterpret_cast<char*>(&length), sizeof(uint32_t));
219     length = ntohl(length);
220     buffer.clear();
221     buffer.resize(length);
222     socket.receive(buffer.data(), length);
223     PlayerData data = protocol.processAddPlayerData(&buffer);
224     currentUpdate.push(std::unique_ptr<UpdateEvent>(new UpdateGUI(std::move(data
225         ))));
226 }

```

jul 21, 20 15:47

UpdateManager.h

Page 1/1

```

1  //
2  // Created by marcos on 13/7/20.
3  //
4
5  #ifndef ARGENTUM_UPDATEMANAGER_H
6  #define ARGENTUM_UPDATEMANAGER_H
7
8  #include "Update.h"
9  #include "../UpdateEvents/UpdateEvent.h"
10 #include <mutex>
11 #include <memory>
12
13 /*Almacena los udpates que recibe del server, los cuales serian consumidos
14  * por el thread principal en el main game loop*/
15
16 class UpdateManager {
17 private:
18     std::mutex m;
19     std::queue<Update> updates;
20
21 public:
22     /*Agrega un update*/
23     void push(Update& update);
24
25     /*Retorna el siguiente update, eliminandolo de la cola de updates*/
26     Update pop();
27
28     /*Retorna la cantidad de updates disponibles (cantidad en la cola)*/
29     int updatesAvailable();
30 };
31
32
33 #endif //ARGENTUM_UPDATEMANAGER_H

```

jul 21, 20 15:47

UpdateManager.cpp

Page 1/1

```

1  //
2  // Created by marcos on 13/7/20.
3  //
4
5  #include "UpdateManager.h"
6  #include "../libs/TPEException.h"
7
8  void UpdateManager::push(Update& update) {
9      std::lock_guard<std::mutex> l(m);
10     updates.emplace(std::move(update));
11 }
12
13 Update UpdateManager::pop() {
14     std::lock_guard<std::mutex> l(m);
15     if (!updates.empty()) {
16         auto update = std::move(updates.front());
17         updates.pop();
18         return update;
19     }
20     throw TPEException("An update was requested from an empty queue!");
21 }
22
23 int UpdateManager::updatesAvailable() {
24     std::lock_guard<std::mutex> l(m);
25     return updates.size();
26 }

```

jul 21, 20 15:47

Update.h

Page 1/1

```

1  //
2  // Created by marcos on 6/29/20.
3  //
4
5  #ifndef ARGENTUM_UPDATE_H
6  #define ARGENTUM_UPDATE_H
7
8  #include <queue>
9  #include <memory>
10 #include "../UpdateEvents/UpdateEvent.h"
11
12 class Update {
13 private:
14     std::queue<std::unique_ptr<UpdateEvent>> events;
15
16 public:
17     Update() = default;
18     Update(Update^ other) noexcept;
19     void push(std::unique_ptr<UpdateEvent> element);
20     std::unique_ptr<UpdateEvent> pop();
21     bool empty();
22 };
23
24 #endif //ARGENTUM_UPDATE_H

```

jul 21, 20 15:47

Update.cpp

Page 1/1

```

1  //
2  // Created by marcos on 19/7/20.
3  //
4
5  #include "Update.h"
6
7  void Update::push(std::unique_ptr<UpdateEvent> element) {
8      events.push(std::move(element));
9  }
10
11 std::unique_ptr<UpdateEvent> Update::pop() {
12     std::unique_ptr<UpdateEvent> element = std::move(events.front());
13     events.pop();
14     return element;
15 }
16
17 bool Update::empty() {
18     return events.empty();
19 }
20
21 Update::Update(Update^ other) noexcept {
22     events = std::move(other.events);
23 }

```

jul 21, 20 15:47

ProtocolEnumTranslator.h

Page 1/2

```

1  //
2  // Created by ivan on 25/6/20.
3  //
4
5  #ifndef ARGENTUM_PROTOCOLENUMTRANSLATOR_H
6  #define ARGENTUM_PROTOCOLENUMTRANSLATOR_H
7
8  #include "../libs/GameEnums.h"
9  #include "../Texture/TextureRepository.h"
10 #include <unordered_map>
11
12 struct FloorTypeTexture{
13     TextureID texture;
14     int index;
15 };
16
17 class ProtocolEnumTranslator {
18 private:
19     std::unordered_map<GameType::Entity,TextureID> entitiesMap;
20     std::unordered_map<GameType::FloorType,FloorTypeTexture> floorTypesMap;
21     std::unordered_map<GameType::Structure,TextureID> structuresMap;
22     std::unordered_map<GameType::Race,TextureID> racesMap;
23     std::unordered_map<GameType::Weapon,TextureID> weaponsMap;
24     std::unordered_map<GameType::Weapon,TextureID> weaponDropsMap;
25     std::unordered_map<GameType::Clothing,TextureID> clothingMap;
26     std::unordered_map<GameType::Clothing,TextureID> clothingDropsMap;
27     std::unordered_map<GameType::Potion,TextureID>potionsMap;
28
29 public:
30     ProtocolEnumTranslator();
31
32     /* Devuelve el id de la textura perteneciente a "entity" */
33     TextureID getEntityTexture(GameType::Entity entity);
34
35     /* Devuelve el id de la textura perteneciente a "floorType". Tambien recibe
36     el
37         * indice para saber que tile de la textura usar */
38     FloorTypeTexture getFloorTypeTexture(GameType::FloorType floorType);
39
40     /* Devuelve el id de la textura perteneciente a "structure" */
41     TextureID getStructureTexture(GameType::Structure structure);
42
43     /* Devuelve el id de la textura perteneciente a la raza "race" */
44     TextureID getRaceTexture(GameType::Race race);
45
46     /* Devuelve el id de la textura perteneciente a "weapon". Esta es la textura
47     * que se usa para mostrar equipada en el jugador */
48     TextureID getWeaponTexture(GameType::Weapon weapon);
49
50     /* Devuelve el id de la textura perteneciente a "weapon". Esta es la textura
51     * que se usa para mostrar como drop y en el inventario */
52     TextureID getWeaponDropTexture(GameType::Weapon weapon);
53
54     /* Devuelve el id de la textura perteneciente a la vestimenta "clothing". Es
55     ta es la textura
56     * que se usa para mostrar equipada en el jugador */
57     TextureID getClothingTexture(GameType::Clothing clothing);
58
59     /* Devuelve el id de la textura perteneciente a "clothing". Esta es la textu
60     ra
61     * que se usa para mostrar como drop y en el inventario */
62     TextureID getClothingDropTexture(GameType::Clothing clothing);
63
64     /* Devuelve el id de la textura perteneciente a "potion" */
65     TextureID getPotionTexture(GameType::Potion potion);
66

```

jul 21, 20 15:47

ProtocolEnumTranslator.h

Page 2/2

```

64     ~ProtocolEnumTranslator();
65
66 private:
67     void _translateEntities();
68     void _translateFloorTypes();
69     void _translateStructures();
70     void _translateRaces();
71     void _translateWeapons();
72     void _translateWeaponDrops();
73     void _translateClothing();
74     void _translateClothingDrops();
75     void _translatePotions();
76 };
77
78 #endif //ARGENTUM_PROTOCOLENUMTRANSLATOR_H
79

```

jul 21, 20 15:47	ProtocolEnumTranslator.cpp	Page 1/4
1	//	
2	// Created by ivan on 25/6/20.	
3	//	
4		
5	#include "ProtocolEnumTranslator.h"	
6		
7	ProtocolEnumTranslator::ProtocolEnumTranslator() {	
8	_translateEntities();	
9	_translateFloorTypes();	
10	_translateStructures();	
11	_translateRaces();	
12	_translateWeapons();	
13	_translateWeaponDrops();	
14	_translateClothing();	
15	_translateClothingDrops();	
16	_translatePotions();	
17	}	
18		
19	void ProtocolEnumTranslator::_translateEntities(){	
20	entitiesMap.emplace(GameType::Entity::SKELETON, Skeleton);	
21	entitiesMap.emplace(GameType::Entity::ZOMBIE, Zombie);	
22	entitiesMap.emplace(GameType::Entity::SPIDER, Spider);	
23	entitiesMap.emplace(GameType::Entity::GOBLIN, Goblin);	
24	entitiesMap.emplace(GameType::Entity::BANKER, Banker);	
25	entitiesMap.emplace(GameType::Entity::GUARD, Guard);	
26	entitiesMap.emplace(GameType::Entity::TRADER, Trader);	
27	entitiesMap.emplace(GameType::Entity::PRIEST, Priest);	
28	entitiesMap.emplace(GameType::Entity::NOTHING, Nothing);	
29	}	
30		
31	void ProtocolEnumTranslator::_translateFloorTypes() {	
32	floorTypesMap.emplace(GameType::FloorType::GRASS0, FloorTypeTexture{Grass, 0})	
33	;	
34	floorTypesMap.emplace(GameType::FloorType::GRASS1, FloorTypeTexture{Grass, 1})	
35	};	
36	floorTypesMap.emplace(GameType::FloorType::GRASS2, FloorTypeTexture{Grass, 2})	
37	};	
38	floorTypesMap.emplace(GameType::FloorType::GRASS3, FloorTypeTexture{Grass, 3})	
39	};	
40	floorTypesMap.emplace(GameType::FloorType::SAND, FloorTypeTexture{Sand, 0});	
41	floorTypesMap.emplace(GameType::FloorType::WATER0, FloorTypeTexture{Water, 0})	
42	};	
43	floorTypesMap.emplace(GameType::FloorType::WATER1, FloorTypeTexture{Water, 1})	
44	};	
45	floorTypesMap.emplace(GameType::FloorType::WATER2, FloorTypeTexture{Water, 2})	
46	};	
47	floorTypesMap.emplace(GameType::FloorType::WATER3, FloorTypeTexture{Water, 3})	
48	};	
49	floorTypesMap.emplace(GameType::FloorType::PRETTY_ROAD0, FloorTypeTexture{PrettyRoad, 0});	
50	floorTypesMap.emplace(GameType::FloorType::PRETTY_ROAD1, FloorTypeTexture{PrettyRoad, 1});	
51	floorTypesMap.emplace(GameType::FloorType::PRETTY_ROAD2, FloorTypeTexture{PrettyRoad, 2});	
52	floorTypesMap.emplace(GameType::FloorType::PRETTY_ROAD3, FloorTypeTexture{PrettyRoad, 3});	
53	floorTypesMap.emplace(GameType::FloorType::PRETTY_GRASS0, FloorTypeTexture{PrettyGrass, 0});	
54	floorTypesMap.emplace(GameType::FloorType::PRETTY_GRASS1, FloorTypeTexture{PrettyGrass, 1});	
55	floorTypesMap.emplace(GameType::FloorType::PRETTY_GRASS2, FloorTypeTexture{PrettyGrass, 2});	
56	floorTypesMap.emplace(GameType::FloorType::PRETTY_GRASS3, FloorTypeTexture{PrettyGrass, 3});	
57	floorTypesMap.emplace(GameType::FloorType::DEAD_GRASS0, FloorTypeTexture{DeadGrass, 0});	

jul 21, 20 15:47	ProtocolEnumTranslator.cpp	Page 2/4
50	floorTypesMap.emplace(GameType::FloorType::DEAD_GRASS1, FloorTypeTexture{DeadGrass, 1});	
51	floorTypesMap.emplace(GameType::FloorType::DEAD_GRASS2, FloorTypeTexture{DeadGrass, 2});	
52	floorTypesMap.emplace(GameType::FloorType::DEAD_GRASS3, FloorTypeTexture{DeadGrass, 3});	
53	floorTypesMap.emplace(GameType::FloorType::DARK_WATER0, FloorTypeTexture{DarkWater, 0});	
54	floorTypesMap.emplace(GameType::FloorType::DARK_WATER1, FloorTypeTexture{DarkWater, 1});	
55	floorTypesMap.emplace(GameType::FloorType::DARK_WATER2, FloorTypeTexture{DarkWater, 2});	
56	floorTypesMap.emplace(GameType::FloorType::DARK_WATER3, FloorTypeTexture{DarkWater, 3});	
57	}	
58		
59	void ProtocolEnumTranslator::_translateStructures() {	
60	structuresMap.emplace(GameType::Structure::BONE_GUY, BoneGuy);	
61	structuresMap.emplace(GameType::Structure::BROKEN_RIP_STONE, BrokenRipStone);	
62	;	
63	structuresMap.emplace(GameType::Structure::BUSH, Bush);	
64	structuresMap.emplace(GameType::Structure::DEAD_BUSH, DeadBush);	
65	structuresMap.emplace(GameType::Structure::DEAD_GUY, DeadGuy);	
66	structuresMap.emplace(GameType::Structure::DEAD_TREE, DeadTree);	
67	structuresMap.emplace(GameType::Structure::FAT_TREE, FatTree);	
68	structuresMap.emplace(GameType::Structure::HANGED_GUY, HangedGuy);	
69	structuresMap.emplace(GameType::Structure::HOUSE1, House1);	
70	structuresMap.emplace(GameType::Structure::HOUSE2, House2);	
71	structuresMap.emplace(GameType::Structure::HOUSE3, House3);	
72	structuresMap.emplace(GameType::Structure::LONG_TREE, LongTree);	
73	structuresMap.emplace(GameType::Structure::PALM_TREE, PalmTree);	
74	structuresMap.emplace(GameType::Structure::RIP_STONE, RipStone);	
75	structuresMap.emplace(GameType::Structure::TREE, Tree);	
76	structuresMap.emplace(GameType::Structure::VERY_DEAD_GUY, VeryDeadGuy);	
77	structuresMap.emplace(GameType::Structure::SUNKEN_COLUMN, SunkenColumn);	
78	structuresMap.emplace(GameType::Structure::SUNKEN_SHIP, SunkenShip);	
79	structuresMap.emplace(GameType::Structure::NO_STRUCTURE, Nothing);	
80	}	
81		
82	void ProtocolEnumTranslator::_translateRaces() {	
83	racesMap.emplace(GameType::Race::HUMAN, HumanHead);	
84	racesMap.emplace(GameType::Race::ELF, ElfHead);	
85	racesMap.emplace(GameType::Race::DWARF, DwarfHead);	
86	racesMap.emplace(GameType::Race::GNOME, GnomeHead);	
87	}	
88		
89	void ProtocolEnumTranslator::_translateWeapons() {	
90	weaponsMap.emplace(GameType::Weapon::LONGSWORD, LongSword);	
91	weaponsMap.emplace(GameType::Weapon::AXE, Axe);	
92	weaponsMap.emplace(GameType::Weapon::WARHAMMER, WarHammer);	
93	weaponsMap.emplace(GameType::Weapon::ASH_ROD, AshRod);	
94	weaponsMap.emplace(GameType::Weapon::ELVEN_FLUTE, Nothing);	
95	weaponsMap.emplace(GameType::Weapon::LINKED_STAFF, LinkedStaff);	
96	weaponsMap.emplace(GameType::Weapon::SIMPLE_BOW, SimpleBow);	
97	weaponsMap.emplace(GameType::Weapon::COMPOSITE_BOW, CompositeBow);	
98	weaponsMap.emplace(GameType::Weapon::GNARLED_STAFF, GnarledStaff);	
99	weaponsMap.emplace(GameType::Weapon::FIST, Nothing);	
100	}	
101		
102	void ProtocolEnumTranslator::_translateWeaponDrops() {	
103	weaponDropsMap.emplace(GameType::Weapon::LONGSWORD, LongSwordDrop);	
104	weaponDropsMap.emplace(GameType::Weapon::AXE, AxeDrop);	
105	weaponDropsMap.emplace(GameType::Weapon::WARHAMMER, WarHammerDrop);	
106	weaponDropsMap.emplace(GameType::Weapon::ASH_ROD, AshRodDrop);	
107	weaponDropsMap.emplace(GameType::Weapon::ELVEN_FLUTE, ElvenFluteDrop);	

jul 21, 20 15:47

ProtocolEnumTranslator.cpp

Page 3/4

```

108 weaponDropsMap.emplace(GameType::Weapon::LINKED_STAFF, LinkedStaffDrop);
109 weaponDropsMap.emplace(GameType::Weapon::SIMPLE_BOW, SimpleBowDrop);
110 weaponDropsMap.emplace(GameType::Weapon::COMPOSITE_BOW, CompositeBowDrop);
111 weaponDropsMap.emplace(GameType::Weapon::GNARLED_STAFF, GnarledStaffDrop);
112 weaponDropsMap.emplace(GameType::Weapon::FIST, Nothing);
113 }
114
115 void ProtocolEnumTranslator::_translateClothing() {
116     clothingMap.emplace(GameType::Clothing::COMMON_CLOTHING, CommonClothing);
117     clothingMap.emplace(GameType::Clothing::LEATHER_ARMOR, LeatherArmor);
118     clothingMap.emplace(GameType::Clothing::PLATE_ARMOR, PlateArmor);
119     clothingMap.emplace(GameType::Clothing::KING_ARMOR, KingArmor);
120     clothingMap.emplace(GameType::Clothing::BLUE_TUNIC, BlueTunic);
121     clothingMap.emplace(GameType::Clothing::HOOD, Hood);
122     clothingMap.emplace(GameType::Clothing::IRON_HELMET, IronHelmet);
123     clothingMap.emplace(GameType::Clothing::TURTLE_SHIELD, TurtleShield);
124     clothingMap.emplace(GameType::Clothing::IRON_SHIELD, IronShield);
125     clothingMap.emplace(GameType::Clothing::MAGIC_HAT, MagicHat);
126     clothingMap.emplace(GameType::Clothing::NO_HELMET, Nothing);
127     clothingMap.emplace(GameType::Clothing::NO_SHIELD, Nothing);
128 }
129
130 void ProtocolEnumTranslator::_translateClothingDrops() {
131     clothingDropsMap.emplace(GameType::Clothing::COMMON_CLOTHING, CommonClothingDrop);
132     clothingDropsMap.emplace(GameType::Clothing::LEATHER_ARMOR, LeatherArmorDrop);
133     clothingDropsMap.emplace(GameType::Clothing::PLATE_ARMOR, PlateArmorDrop);
134     clothingDropsMap.emplace(GameType::Clothing::KING_ARMOR, KingArmorDrop);
135     clothingDropsMap.emplace(GameType::Clothing::BLUE_TUNIC, BlueTunicDrop);
136     clothingDropsMap.emplace(GameType::Clothing::HOOD, HoodDrop);
137     clothingDropsMap.emplace(GameType::Clothing::IRON_HELMET, IronHelmetDrop);
138     clothingDropsMap.emplace(GameType::Clothing::TURTLE_SHIELD, TurtleShieldDrop);
139     clothingDropsMap.emplace(GameType::Clothing::IRON_SHIELD, IronShieldDrop);
140     clothingDropsMap.emplace(GameType::Clothing::MAGIC_HAT, MagicHatDrop);
141     clothingDropsMap.emplace(GameType::Clothing::NO_HELMET, Nothing);
142     clothingDropsMap.emplace(GameType::Clothing::NO_SHIELD, Nothing);
143 }
144
145 void ProtocolEnumTranslator::_translatePotions() {
146     potionsMap.emplace(GameType::Potion::HEALTH_POTION, HealthPotion);
147     potionsMap.emplace(GameType::Potion::MANA_POTION, ManaPotion);
148 }
149
150 TextureID ProtocolEnumTranslator::getEntityTexture(GameType::Entity entity) {
151     return entitiesMap.at(entity);
152 }
153
154 TextureID ProtocolEnumTranslator::getStructureTexture(GameType::Structure structure) {
155     return structuresMap.at(structure);
156 }
157 TextureID ProtocolEnumTranslator::getRaceTexture(GameType::Race race) {
158     return racesMap.at(race);
159 }
160 TextureID ProtocolEnumTranslator::getWeaponTexture(GameType::Weapon weapon) {
161     return weaponsMap.at(weapon);
162 }
163 TextureID ProtocolEnumTranslator::getWeaponDropTexture(GameType::Weapon weapon) {
164     return weaponDropsMap.at(weapon);
165 }
166 TextureID ProtocolEnumTranslator::getClothingTexture(GameType::Clothing clothing) {
167     return clothingMap.at(clothing);
168 }

```

jul 21, 20 15:47

ProtocolEnumTranslator.cpp

Page 4/4

```

169 TextureID ProtocolEnumTranslator::getClothingDropTexture(GameType::Clothing clothing) {
170     return clothingDropsMap.at(clothing);
171 }
172 TextureID ProtocolEnumTranslator::getPotionTexture(GameType::Potion potion) {
173     return potionsMap.at(potion);
174 }
175
176 FloorTypeTexture ProtocolEnumTranslator::getFloorTypeTexture(GameType::FloorType floorType) {
177     return floorTypesMap.at(floorType);
178 }
179
180 ProtocolEnumTranslator::~ProtocolEnumTranslator() = default;

```


jul 21, 20 15:47

clientMain.cpp

Page 1/1

```

1  #include "Client/ArgentumClientSide.h"
2  #include "../libs/TPException.h"
3  #include <iostream>
4
5  int main(int argc, char** argv) {
6      try {
7          ArgentumClientSide::run(argc);
8      } catch (TPException& e) {
9          std::cerr << e.what() << " in Client!" << std::endl;
10     } catch (...) {
11         std::cerr << "Unknown error in Client!" << std::endl;
12     }
13 }

```

jul 21, 20 15:47

WithdrawCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #ifndef ARGENTUM_WITHDRAWCOMMAND_H
6  #define ARGENTUM_WITHDRAWCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class WithdrawCommand : public InputCommand{
12 private:
13     Coordinate tile;
14     std::string item;
15
16 public:
17     explicit WithdrawCommand(Coordinate tile, std::string^ item) : tile(tile),
18         item(item) {}
19     void operator()(std::stringstream& msgBuffer) override;
20 };
21
22 #endif //ARGENTUM_WITHDRAWCOMMAND_H

```

jul 21, 20 15:47

WithdrawCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #include "WithdrawCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9
10 void WithdrawCommand::operator()(std::stringstream &msgBuffer) {
11     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_WITHDRAW)
12     ;
13     msgpack::type::tuple<std::string, int32_t, int32_t> depositInfo;
14     depositInfo = {item, tile.i, tile.j};
15     msgpack::pack(msgBuffer, event);
16     msgpack::pack(msgBuffer, depositInfo);
17 }

```

jul 21, 20 15:47

SellCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_SELLCOMMAND_H
6  #define ARGENTUM_SELLCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class SellCommand : public InputCommand {
12 private:
13     Coordinate tile;
14     std::string item;
15
16 public:
17     explicit SellCommand(Coordinate tile, std::string^ item) : tile(tile), item
18     (item) {}
19     void operator()(std::stringstream& msgBuffer) override;
20 };
21
22
23 #endif // ARGENTUM_SELLCOMMAND_H

```

jul 21, 20 15:47

SellCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #include "SellCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9
10 void SellCommand::operator()(std::stringstream &msgBuffer) {
11     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_SELL);
12     msgpack::type::tuple<std::string, int32_t, int32_t> saleInfo;
13     saleInfo = {item, tile.i, tile.j};
14     msgpack::pack(msgBuffer, event);
15     msgpack::pack(msgBuffer, saleInfo);
16 }

```

jul 21, 20 15:47

ResurrectCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_RESURRECTCOMMAND_H
6  #define ARGENTUM_RESURRECTCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class ResurrectCommand : public InputCommand {
12 private:
13     Coordinate tile;
14
15 public:
16     explicit ResurrectCommand(Coordinate tile) : tile(tile) {}
17     void operator()(std::stringstream& msgBuffer) override;
18 };
19
20 #endif // ARGENTUM_RESURRECTCOMMAND_H

```

jul 21, 20 15:47

ResurrectCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #include "ResurrectCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9
10 void ResurrectCommand::operator()(std::stringstream &msgBuffer) {
11     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_RESURRECT
12 );
13     msgpack::type::tuple<int32_t, int32_t> selectedTile;
14     selectedTile = {tile.i, tile.j};
15     msgpack::pack(msgBuffer, event);
16     msgpack::pack(msgBuffer, selectedTile);
17 }

```

jul 21, 20 15:47

RequestInventoryNamesCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 13/7/20.
3  //
4
5  #ifndef ARGENTUM_REQUESTINVENTORYNAMESCOMMAND_H
6  #define ARGENTUM_REQUESTINVENTORYNAMESCOMMAND_H
7
8  #include "InputCommand.h"
9
10 class RequestInventoryNamesCommand : public InputCommand{
11 public:
12     void operator()(std::stringstream& msgBuffer) override;
13 };
14
15
16 #endif //ARGENTUM_REQUESTINVENTORYNAMESCOMMAND_H

```

jul 21, 20 15:47

RequestInventoryNamesCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 13/7/20.
3  //
4
5  #include "RequestInventoryNamesCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9  void RequestInventoryNamesCommand::operator()(std::stringstream &msgBuffer) {
10     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_REQUEST_I
11     NVENTORY_NAMES);
12     msgpack::pack(msgBuffer, event);
13 }

```

jul 21, 20 15:47

PickUpCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_PICKUPCOMMAND_H
6  #define ARGENTUM_PICKUPCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class PickUpCommand : public InputCommand{
12 public:
13     void operator()(std::stringstream& msgBuffer) override;
14 };
15
16
17 #endif //ARGENTUM_PICKUPCOMMAND_H

```

jul 21, 20 15:47

PickUpCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #include "PickUpCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9  void PickUpCommand::operator()(std::stringstream &msgBuffer) {
10     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_PICK_UP);
11     msgpack::pack(msgBuffer, event);
12 }

```

jul 21, 20 15:47

MessageToPlayerCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #ifndef ARGENTUM_MESSAGETOPLAYERCOMMAND_H
6  #define ARGENTUM_MESSAGETOPLAYERCOMMAND_H
7  #include "InputCommand.h"
8  #include "../Map/Coordinate.h"
9
10 class MessageToPlayerCommand : public InputCommand{
11 private:
12     std::string nickname, msg;
13 public:
14     MessageToPlayerCommand(std::string^ nickname, std::string^ msg) : nickname(
15         nickname),
16         msg(msg) {}
17     void operator()(std::stringstream& msgBuffer) override;
18 };
19
20
21 #endif // ARGENTUM_MESSAGETOPLAYERCOMMAND_H

```

jul 21, 20 15:47

MessageToPlayerCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #include "MessageToPlayerCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9
10 void MessageToPlayerCommand::operator()(std::stringstream &msgBuffer) {
11     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_SEND_MSG)
12     ;
13     msgpack::type::tuple<std::string, std::string> msgInfo;
14     msgInfo = {nickname, msg};
15     msgpack::pack(msgBuffer, event);
16     msgpack::pack(msgBuffer, msgInfo);
17 }

```

jul 21, 20 15:47

MeditateCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 7/7/20.
3  //
4
5  #ifndef ARGENTUM_MEDITATECOMMAND_H
6  #define ARGENTUM_MEDITATECOMMAND_H
7
8  #include "InputCommand.h"
9
10 class MeditateCommand : public InputCommand{
11 public:
12     void operator()(std::stringstream& msgBuffer) override;
13 };
14
15 #endif //ARGENTUM_MEDITATECOMMAND_H

```

jul 21, 20 15:47

MeditateCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 7/7/20.
3  //
4
5  #include "MeditateCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9
10 void MeditateCommand::operator()(std::stringstream& msgBuffer) {
11     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_MEDITATE)
12     ;
13     msgpack::pack(msgBuffer, event);
14 }

```

jul 21, 20 15:47

ListCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_LISTCOMMAND_H
6  #define ARGENTUM_LISTCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class ListCommand : public InputCommand{
12 private:
13     Coordinate tile;
14
15 public:
16     explicit ListCommand(Coordinate tile) : tile(tile) {}
17     void operator()(std::stringstream& msgBuffer) override;
18 };
19
20
21 #endif //ARGENTUM_LISTCOMMAND_H

```


jul 21, 20 15:47

ListCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #include "ListCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9  void ListCommand::operator()(std::stringstream &msgBuffer) {
10     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_LIST);
11     msgpack::type::tuple<int32_t, int32_t> selectedTile;
12     selectedTile = {tile.i, tile.j};
13     msgpack::pack(msgBuffer, event);
14     msgpack::pack(msgBuffer, selectedTile);
15 }

```

jul 21, 20 15:47

InputCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 7/7/20.
3  //
4
5  #ifndef ARGENTUM_INPUTCOMMAND_H
6  #define ARGENTUM_INPUTCOMMAND_H
7
8  #include <sstream>
9  #include <msgpack.hpp>
10 #include "../libs/GameEnums.h"
11
12 // Interfaz de los comandos del minichat.
13
14 class InputCommand {
15 public:
16     virtual void operator()(std::stringstream& msgBuffer) = 0;
17     virtual ~InputCommand() = default;
18 };
19
20 #endif // ARGENTUM_INPUTCOMMAND_H

```

jul 21, 20 15:47

HealCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #ifndef ARGENTUM_HEALCOMMAND_H
6  #define ARGENTUM_HEALCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class HealCommand : public InputCommand{
12 private:
13     Coordinate tile;
14
15 public:
16     explicit HealCommand(Coordinate tile) : tile(tile) {}
17     void operator()(std::stringstream& msgBuffer) override;
18 };
19
20
21 #endif //ARGENTUM_HEALCOMMAND_H

```

jul 21, 20 15:47

HealCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #include "HealCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9
10 void HealCommand::operator()(std::stringstream &msgBuffer) {
11     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_HEAL);
12     msgpack::type::tuple<int32_t, int32_t> selectedTile;
13     selectedTile = {tile.i, tile.j};
14     msgpack::pack(msgBuffer, event);
15     msgpack::pack(msgBuffer, selectedTile);
16 }

```

jul 21, 20 15:47

DropCommand.h

Page 1/1

```
1 //
2 // Created by ivan on 9/7/20.
3 //
4
5 #ifndef ARGENTUM_DROPCOMMAND_H
6 #define ARGENTUM_DROPCOMMAND_H
7
8 #include "InputCommand.h"
9 #include "../Map/Coordinate.h"
10
11 class DropCommand : public InputCommand{
12 private:
13     int32_t slot;
14 public:
15     explicit DropCommand(int32_t slot) : slot(slot) {}
16     void operator()(std::stringstream& msgBuffer) override;
17 };
18
19
20 #endif //ARGENTUM_DROPCOMMAND_H
```

jul 21, 20 15:47

DropCommand.cpp

Page 1/1

```
1 //
2 // Created by ivan on 9/7/20.
3 //
4
5 #include "DropCommand.h"
6
7 MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9 void DropCommand::operator()(std::stringstream &msgBuffer) {
10     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_DROP);
11     msgpack::type::tuple<int32_t> inventorySlot;
12     inventorySlot = slot;
13     msgpack::pack(msgBuffer, event);
14     msgpack::pack(msgBuffer, inventorySlot);
15 }
```

jul 21, 20 15:47

DepositCommand.h

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #ifndef ARGENTUM_DEPOSITCOMMAND_H
6  #define ARGENTUM_DEPOSITCOMMAND_H
7
8  #include "InputCommand.h"
9  #include "../Map/Coordinate.h"
10
11 class DepositCommand : public InputCommand{
12 private:
13     Coordinate tile;
14     std::string item;
15
16 public:
17     explicit DepositCommand(Coordinate tile, std::string^ item) : tile(tile), i
18 tem(item) {}
19     void operator()(std::stringstream& msgBuffer) override;
20 };
21
22 #endif //ARGENTUM_DEPOSITCOMMAND_H

```

jul 21, 20 15:47

DepositCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 10/7/20.
3  //
4
5  #include "DepositCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9  void DepositCommand::operator()(std::stringstream &msgBuffer) {
10     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_DEPOSIT);
11     msgpack::type::tuple<std::string, int32_t, int32_t> depositInfo;
12     depositInfo = {item, tile.i, tile.j};
13     msgpack::pack(msgBuffer, event);
14     msgpack::pack(msgBuffer, depositInfo);
15 }

```

jul 21, 20 15:47

CommandVerifier.h

Page 1/1

```

1  //
2  // Created by ivan on 7/7/20.
3  //
4
5  #ifndef ARGENTUM_COMMANDVERIFIER_H
6  #define ARGENTUM_COMMANDVERIFIER_H
7
8  #include "../libs/GameEnums.h"
9  #include "InputCommand.h"
10 #include <memory>
11 #include <unordered_map>
12
13 class GameGUI;
14
15 class CommandVerifier {
16 private:
17     std::unordered_map<std::string, GameType::PlayerEvent> commands;
18     std::string input;
19
20 public:
21     /* Constructor */
22     CommandVerifier();
23     /* Verifica que comando ingrese y devuelve el functor */
24     std::unique_ptr<InputCommand> verifyCommand(GameGUI& game, std::string& cmd
25 );
26
27 private:
28     void _initCommands();
29     std::unique_ptr<InputCommand> _processMeditate();
30     std::unique_ptr<InputCommand> _processPickUp();
31     std::unique_ptr<InputCommand> _processDrop(GameGUI& game);
32     std::unique_ptr<InputCommand> _processList(GameGUI &game);
33     std::unique_ptr<InputCommand> _processResurrect(GameGUI &game);
34     std::unique_ptr<InputCommand> _processSell(GameGUI &game);
35     std::unique_ptr<InputCommand> _processBuy(GameGUI &game);
36     std::unique_ptr<InputCommand> _processHeal(GameGUI& game);
37     std::unique_ptr<InputCommand> _processDeposit(GameGUI &game);
38     std::unique_ptr<InputCommand> _processWithdraw(GameGUI &game);
39     std::unique_ptr<InputCommand> _processSendMessageToPlayer();
40     std::unique_ptr<InputCommand> _processRequestInventoryNames();
41
42     static void _processGold(std::string &parameter);
43 };
44
45
46 #endif //ARGENTUM_COMMANDVERIFIER_H

```

jul 21, 20 15:47

CommandVerifier.cpp

Page 1/4

```

1  //
2  // Created by ivan on 7/7/20.
3  //
4
5  #include <sstream>
6  #include "CommandVerifier.h"
7  #include "MeditateCommand.h"
8  #include "PickUpCommand.h"
9  #include "DropCommand.h"
10 #include "ListCommand.h"
11 #include "SellCommand.h"
12 #include "BuyCommand.h"
13 #include "ResurrectCommand.h"
14 #include "HealCommand.h"
15 #include "DepositCommand.h"
16 #include "WithdrawCommand.h"
17 #include "MessageToPlayerCommand.h"
18 #include "RequestInventoryNamesCommand.h"
19 #include "../GameGUI.h"
20
21 CommandVerifier::CommandVerifier() {
22     _initCommands();
23 }
24
25 /* Inicializa el unordered_map de comandos */
26 void CommandVerifier::_initCommands() {
27     commands.emplace("/meditate", GameType::PLAYER_MEDITATE);
28     commands.emplace("/revive", GameType::PLAYER_RESURRECT);
29     commands.emplace("/heal", GameType::PLAYER_HEAL);
30     commands.emplace("/deposit", GameType::PLAYER_DEPOSIT);
31     commands.emplace("/withdraw", GameType::PLAYER_WITHDRAW);
32     commands.emplace("/list", GameType::PLAYER_LIST);
33     commands.emplace("/buy", GameType::PLAYER_BUY);
34     commands.emplace("/sell", GameType::PLAYER_SELL);
35     commands.emplace("/take", GameType::PLAYER_PICK_UP);
36     commands.emplace("/drop", GameType::PLAYER_DROP);
37     commands.emplace("/inventory", GameType::PLAYER_REQUEST_INVENTORY_NAMES);
38 }
39
40 std::unique_ptr<InputCommand> CommandVerifier::verifyCommand(GameGUI& game,
41     std::string& inputCmd) {
42     std::unique_ptr<InputCommand> command;
43     input = inputCmd;
44
45     //Agarro lo que tenga antes de un espacio. Eso deberia ser el comando
46     std::string cmd = input.substr(0, input.find(' ', 0));
47     GameType::PlayerEvent event;
48     if (cmd.front() == '@') { //Antes de ver si es un comando veo si es un nickna
49         me
50         command = _processSendMessageToPlayer();
51     } else {
52         try {
53             event = commands.at(cmd);
54             switch (event) {
55                 case GameType::PLAYER_PICK_UP:
56                     command = _processPickUp();
57                     break;
58                 case GameType::PLAYER_DROP:
59                     command = _processDrop(game);
60                     break;
61                 case GameType::PLAYER_LIST:
62                     command = _processList(game);
63                     break;
64                 case GameType::PLAYER_RESURRECT:
65                     command = _processResurrect(game);
66                     break;

```

jul 21, 20 15:47

CommandVerifier.cpp

Page 2/4

```

66         case GameType::PLAYER_HEAL:
67             command = _processHeal(game);
68             break;
69         case GameType::PLAYER_BUY:
70             command = _processBuy(game);
71             break;
72         case GameType::PLAYER_SELL:
73             command = _processSell(game);
74             break;
75         case GameType::PLAYER_WITHDRAW:
76             command = _processWithdraw(game);
77             break;
78         case GameType::PLAYER_DEPOSIT:
79             command = _processDeposit(game);
80             break;
81         case GameType::PLAYER_MEDITATE:
82             command = _processMeditate();
83             break;
84         case GameType::PLAYER_REQUEST_INVENTORY_NAMES:
85             command = _processRequestInventoryNames();
86             break;
87         default:
88             break;
89     }
90     } catch (std::exception& e) {
91         //Si no encuentra el comando en el unordered_map es que no es un com
92         ando
93         //valido asi que devuelvo nullptr
94         return nullptr;
95     }
96     }
97     return command;
98 }
99 std::unique_ptr<InputCommand> CommandVerifier::_processRequestInventoryNames() {
100     //Chequeo que no haya nada escrito despues del comando
101     if (input.size() > input.find(' ', 0)) {
102         return nullptr;
103     }
104     return std::unique_ptr<InputCommand>(new RequestInventoryNamesCommand());
105 }
106
107 std::unique_ptr<InputCommand> CommandVerifier::_processMeditate() {
108     //Chequeo que no haya nada escrito despues del comando
109     if (input.size() > input.find(' ', 0)) {
110         return nullptr;
111     }
112     return std::unique_ptr<InputCommand>(new MeditateCommand());
113 }
114
115 std::unique_ptr<InputCommand> CommandVerifier::_processPickUp() {
116     //Chequeo que no haya nada escrito despues del comando
117     if (input.size() > input.find(' ', 0)) {
118         return nullptr;
119     }
120     return std::unique_ptr<InputCommand>(new PickupCommand());
121 }
122
123 std::unique_ptr<InputCommand> CommandVerifier::_processDrop(GameGUI& game) {
124     //Chequeo que no haya nada escrito despues del comando
125     if (input.size() > input.find(' ', 0)) {
126         return nullptr;
127     }
128     return std::unique_ptr<InputCommand>(new DropCommand(game.getSelector().getI
129         nventorySlot()));

```

jul 21, 20 15:47

CommandVerifier.cpp

Page 3/4

```

130
131 std::unique_ptr<InputCommand> CommandVerifier::_processList(GameGUI& game) {
132     //Chequeo que no haya nada escrito despues del comando
133     if (input.size() > input.find(' ', 0)) {
134         return nullptr;
135     }
136     return std::unique_ptr<InputCommand>(new ListCommand(game.getSelector().getS
137         electedTile()));
138 }
139
140 std::unique_ptr<InputCommand> CommandVerifier::_processResurrect(GameGUI& game)
141 {
142     //Chequeo que no haya nada escrito despues del comando
143     if (input.size() > input.find(' ', 0)) {
144         return nullptr;
145     }
146     return std::unique_ptr<InputCommand>(new ResurrectCommand(game.getSelector()
147         .getSelectedTile()));
148 }
149
150 std::unique_ptr<InputCommand> CommandVerifier::_processHeal(GameGUI& game) {
151     //Chequeo que no haya nada escrito despues del comando
152     if (input.size() > input.find(' ', 0)) {
153         return nullptr;
154     }
155     return std::unique_ptr<InputCommand>(new HealCommand(game.getSelector().getS
156         electedTile()));
157 }
158
159 std::unique_ptr<InputCommand> CommandVerifier::_processSell(GameGUI& game) {
160     std::string parameters;
161     if (input.size() > input.find(' ', 0)) {
162         //Agarro lo que haya dsps del espacio que deberia ser el item que quiero
163         vender
164         parameters = input.substr(input.find(' ', 0) + 1, input.size());
165         if (!parameters.empty()) {
166             return std::unique_ptr<InputCommand>(new SellCommand(
167                 game.getSelector().getSelectedTile(), std::move(parameters))
168         );
169         }
170     }
171     return nullptr;
172 }
173
174 std::unique_ptr<InputCommand> CommandVerifier::_processBuy(GameGUI& game) {
175     std::string parameters;
176     if (input.size() > input.find(' ', 0)) {
177         //Agarro lo que haya dsps del espacio que deberian ser los parametros
178         parameters = input.substr(input.find(' ', 0) + 1, input.size());
179         if (!parameters.empty()) {
180             return std::unique_ptr<InputCommand>(new BuyCommand(
181                 game.getSelector().getSelectedTile(), std::move(parameters))
182         );
183         }
184     }
185     return nullptr;
186 }
187
188 std::unique_ptr<InputCommand> CommandVerifier::_processDeposit(GameGUI& game) {
189     std::string parameters;
190     int separator = input.find(' ', 0);
191     if ((int)input.size() > separator ^ separator != -1) {
192         //Agarro lo que haya dsps del espacio que deberian ser los parametros
193         parameters = input.substr(separator + 1, input.size());
194         //Como para el gold tambien necesito una cantidad lo proceso distinto a
195         un item

```

jul 21, 20 15:47

CommandVerifier.cpp

Page 4/4

```

188     if (parameters.find("Gold", 0) != std::string::npos) {
189         _processGold(parameters);
190     }
191     if (!parameters.empty()){
192         return std::unique_ptr<InputCommand>(new DepositCommand(
193             game.getSelector().getSelectedTile(), std::move(parameters))
194     );
195     }
196     return nullptr;
197 }
198
199 std::unique_ptr<InputCommand> CommandVerifier::_processWithdraw(GameGUI& game) {
200     std::string parameters;
201     int separator = input.find(' ', 0);
202     if ((int)input.size() > separator ^ separator != -1) {
203         //Agarro lo que haya dsps del espacio que deberian ser los parametros
204         parameters = input.substr(separator + 1, input.size());
205         //Como para el gold tambien necesito una cantidad lo proceso distinto a
206         un item
207         if (parameters.find("Gold", 0) != std::string::npos) {
208             _processGold(parameters);
209         }
210         if (!parameters.empty()){
211             return std::unique_ptr<InputCommand>(new WithdrawCommand(
212                 game.getSelector().getSelectedTile(), std::move(parameters))
213         );
214     }
215     return nullptr;
216 }
217
218 void CommandVerifier::_processGold(std::string& parameter) {
219     int separator = parameter.find(' ', 0);
220     if ((int)parameter.size() > separator ^ separator != -1) {
221         //Agarro la parte del string que deberia tener la cantidad de gold
222         std::string goldAmount = parameter.substr(parameter.find(' ', 0) + 1,
223             parameter.size());
224         try {
225             std::stoi(goldAmount);
226         } catch (std::exception &e) {
227             parameter = ""; //Si la cantidad no es un numero
228         }
229         } else {
230             parameter = ""; //Si no tengo una cantidad de oro
231         }
232     }
233
234 std::unique_ptr<InputCommand> CommandVerifier::_processSendMessageToPlayer() {
235     int separator = input.find(' ');
236     if ((int)input.size() > separator ^ separator != -1) {
237         std::string nickname = input.substr(1, separator-1);
238         std::string msg = input.substr(separator + 1, input.size());
239         if (!msg.empty()) {
240             return std::unique_ptr<InputCommand>(new MessageToPlayerCommand(
241                 std::move(nickname), std::move(msg)));
242         }
243     }
244     return nullptr;

```

jul 21, 20 15:47

BuyCommand.h

Page 1/1

```

1 //
2 // Created by ivan on 9/7/20.
3 //
4
5 #ifndef ARGENTUM_BUYCOMMAND_H
6 #define ARGENTUM_BUYCOMMAND_H
7
8 #include "InputCommand.h"
9 #include "../Map/Coordinate.h"
10
11 class BuyCommand : public InputCommand{
12 private:
13     Coordinate tile;
14     std::string item;
15
16 public:
17     explicit BuyCommand(Coordinate tile, std::string& item) : tile(tile), item(
18         item) {}
19     void operator()(std::stringstream& msgBuffer) override;
20 };
21
22 #endif // ARGENTUM_BUYCOMMAND_H

```

jul 21, 20 15:47

BuyCommand.cpp

Page 1/1

```

1  //
2  // Created by ivan on 9/7/20.
3  //
4
5  #include "BuyCommand.h"
6
7  MSGPACK_ADD_ENUM(GameType::PlayerEvent)
8
9  void BuyCommand::operator()(std::stringstream &msgBuffer) {
10     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_BUY);
11     msgpack::type::tuple<std::string, int32_t, int32_t> buyInfo;
12     buyInfo = {item, tile.i, tile.j};
13     msgpack::pack(msgBuffer, event);
14     msgpack::pack(msgBuffer, buyInfo);
15 }

```

jul 21, 20 15:47

GameInitializer.h

Page 1/1

```

1  //
2  // Created by marcos on 7/1/20.
3  //
4
5  #ifndef ARGENTUM_GAMEINITIALIZER_H
6  #define ARGENTUM_GAMEINITIALIZER_H
7
8  #include <vector>
9  #include <string>
10 #include "../libs/GameEnums.h"
11
12 class GameGUI;
13 class Socket;
14 class ClientProtocol;
15
16 class GameInitializer {
17 private:
18     GameGUI& game;
19     Socket& socket;
20     ClientProtocol& protocol;
21
22 public:
23     GameInitializer(GameGUI& _game, Socket& _socket, ClientProtocol& _protocol)
24     :
25         game(_game), socket(_socket), protocol(_protocol) {}
26
27     /* Manda al servidor la informacion para crear un nuevo jugador */
28     void createPlayer(const std::string &nickname, GameType::Race race,
29                     GameType::Class _class);
30
31     /* Manda al servidor la informacion para cargar un jugador */
32     void loadPlayer(const std::string &nickname);
33
34     /* Recibe el estado inicial del juego */
35     void initializeGame();
36
37 private:
38     void _receiveMapInfo();
39     void _receiveCurrentGameState();
40     void _loadMap(std::vector<char>& buffer);
41     void _processAddEntity(std::vector<char>& buffer, std::size_t& offset);
42     void _receivePlayerData();
43 };
44
45 #endif //ARGENTUM_GAMEINITIALIZER_H

```


jul 21, 20 15:47

GameInitializer.cpp

Page 1/3

```

1  //
2  // Created by marcos on 7/1/20.
3  //
4
5  #include "GameInitializer.h"
6  #include <stdint>
7  #include "../libs/Socket.h"
8  #include <msgpack.hpp>
9  #include "GameGUI.h"
10 #include "ProtocolEnumTranslator.h"
11 #include "ClientProtocol.h"
12 #include "CitizenData.h"
13
14 MSGPACK_ADD_ENUM(GameType::PlayerEvent)
15 MSGPACK_ADD_ENUM(GameType::EventID)
16 MSGPACK_ADD_ENUM(GameType::Race)
17 MSGPACK_ADD_ENUM(GameType::Class)
18 MSGPACK_ADD_ENUM(GameType::Entity)
19 MSGPACK_ADD_ENUM(GameType::Structure)
20 MSGPACK_ADD_ENUM(GameType::FloorType)
21
22 void GameInitializer::initializeGame() {
23     _receiveMapInfo();
24     _receiveCurrentGameState();
25 }
26
27 /* Recibe la informacion del mapa */
28 void GameInitializer::_receiveMapInfo() {
29     int32_t msgLength;
30     socket.receive((char*)&msgLength, sizeof(msgLength));
31     msgLength = ntohl(msgLength);
32     std::vector<char> buffer(msgLength);
33     socket.receive(buffer.data(), buffer.size());
34     _loadMap(buffer);
35 }
36
37 /* Recibe el estado inicial del juego */
38 void GameInitializer::_receiveCurrentGameState() {
39     int32_t msgLength;
40     socket.receive((char*)&msgLength, sizeof(msgLength));
41     msgLength = ntohl(msgLength);
42     std::vector<char> buffer(msgLength);
43     socket.receive(buffer.data(), msgLength);
44     std::size_t offset = 0;
45     msgpack::object_handle handler;
46
47     while (offset < static_cast<size_t>(msgLength)) {
48         handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
49         msgpack::type::tuple<GameType::EventID> id;
50         handler->convert(id);
51         if (std::get<0>(id) == GameType::CREATE_ITEM) {
52             ItemData data = protocol.processAddItem(&buffer, offset);
53             game.createItem(data.position, data.texture);
54         } else if (std::get<0>(id) == GameType::CREATE_ENTITY) {
55             _processAddEntity(buffer, offset);
56         }
57     }
58     game.getMap().update();
59     _receivePlayerData();
60 }
61
62 /* Procesa la entidad que recibe del server y la agrega al juego */
63 void GameInitializer::_processAddEntity(std::vector<char>& buffer, std::size_t&
offset) {
64     msgpack::object_handle handler = msgpack::unpack(buffer.data(), buffer.size(
), offset);

```

jul 21, 20 15:47

GameInitializer.cpp

Page 2/3

```

65     msgpack::type::tuple<GameType::Entity, std::string, int32_t> entityData;
66     handler->convert(entityData);
67     if (std::get<0>(entityData) != GameType::PLAYER) {
68         EntityData data = protocol.processAddNPC(&buffer, entityData, offset);
69         game.addNPC(data);
70     } else {
71         MapPlayerData data = protocol.processAddPlayer(&buffer, entityData, offs
et);
72         game.addPlayer(data);
73     }
74 }
75
76 /* Carga las texturas de cada tile del mapa */
77 void GameInitializer::_loadMap(std::vector<char>& buffer) {
78     std::size_t offset = 0;
79     msgpack::object_handle handler = msgpack::unpack(buffer.data(), buffer.size(
), offset);
80     msgpack::type::tuple<int32_t, int32_t> mapSize;
81     ProtocolEnumTranslator translator;
82     handler->convert(mapSize);
83     int rows = std::get<0>(mapSize);
84     int columns = std::get<1>(mapSize);
85     game.setMapSize(rows, columns);
86     for (int i = 0; i < rows; ++i) {
87         for (int j = 0; j < columns; ++j) {
88             handler = msgpack::unpack(buffer.data(), buffer.size(), offset);
89             msgpack::type::tuple<GameType::FloorType, GameType::Structure,
GameType::Entity, std::string> tileInfo;
90             handler->convert(tileInfo);
91             CitizenData citizen = {translator.getEntityTexture(std::get<2>(tileI
nfo)),
92                                     std::get<3>(tileInfo)};
93             game.loadTileData({i, j}, translator.getFloorTypeTexture(std::get<0>
(tileInfo)),
94                                     translator.getStructureTexture(std::get<1>(tileInf
o)),
95                                     citizen);
96         }
97     }
98 }
99
100 void GameInitializer::createPlayer(const std::string& nickname, GameType::Race r
ace,
101                                     GameType::Class _class) {
102     game.getMap().setPlayerNickname(nickname); //Para despues poder buscar la pos
icion del player en Map
103     std::stringstream msgBuffer;
104     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::CREATE_PLAYER);
105     msgpack::type::tuple<std::string, GameType::Race, GameType::Class> playerInf
o;
106     playerInfo = {nickname, race, _class};
107     msgpack::pack(msgBuffer, event);
108     msgpack::pack(msgBuffer, playerInfo);
109     std::string aux = msgBuffer.str();
110     uint32_t length = aux.size();
111     length = htonl(aux.size());
112     std::vector<char> sendBuffer(sizeof(uint32_t));
113     ClientProtocol::loadBytes(sendBuffer, &length, sizeof(uint32_t));
114     std::copy(aux.begin(), aux.end(), std::back_inserter(sendBuffer));
115     socket.send(sendBuffer.data(), sendBuffer.size());
116 }
117
118 void GameInitializer::loadPlayer(const std::string& nickname) {
119     game.getMap().setPlayerNickname(nickname); //Para despues poder buscar la pos
icion del player en Map
120     std::stringstream msgBuffer;

```

jul 21, 20 15:47

GameInitializer.cpp

Page 3/3

```

122 msgpack::type::tuple<GameType::PlayerEvent> event(GameType::LOAD_PLAYER);
123 msgpack::type::tuple<std::string> playerInfo;
124 playerInfo = {nickname};
125 msgpack::pack(msgBuffer, event);
126 msgpack::pack(msgBuffer, playerInfo);
127 std::string aux = msgBuffer.str();
128 uint32_t length = aux.size();
129 length = htonl(aux.size());
130 std::vector<char> sendBuffer(sizeof(uint32_t));
131 ClientProtocol::loadBytes(sendBuffer, &length, sizeof(uint32_t));
132 std::copy(aux.begin(), aux.end(), std::back_inserter(sendBuffer));
133 socket.send(sendBuffer.data(), sendBuffer.size());
134 }
135
136 /* Recibe toda la informacion inicial del jugador */
137 void GameInitializer::_receivePlayerData() {
138     uint32_t length = 0;
139     socket.receive(reinterpret_cast<char*>(&length), sizeof(uint32_t));
140     length = ntohl(length);
141     std::vector<char> buffer(length);
142     socket.receive(buffer.data(), length);
143     PlayerData data = protocol.processAddPlayerData(&buffer);
144     game.getPlayerInventory().updateGold(data.generalInfo.gold, data.generalInfo
.safeGold);
145     for (const auto & item : data.equippedItems) {
146         game.getPlayerInventory().addEquipableItem(std::get<0>(item),
147             std::get<1>(item));
148     }
149     for (const auto & item : data.inventoryItems) {
150         game.getPlayerInventory().addInventoryItem(std::get<0>(item),
151             std::get<1>(item));
152     }
153     game.setCameraOn(data.generalInfo.nickname);
154     game.getPlayerInfo().update(data.generalInfo);
155     game.getMinichat().receiveText(data.minichatText);
156 }
157

```

jul 21, 20 15:47

GameGUI.h

Page 1/2

```

1 //
2 // Created by marcos on 6/25/20.
3 //
4
5 #ifndef ARGENTUM_GAMEGUI_H
6 #define ARGENTUM_GAMEGUI_H
7
8 #include "../Screen/Window.h"
9 #include "../Texture/TextureRepository.h"
10 #include "GameConstants.h"
11 #include "../Map/Map.h"
12 #include "../Graphics/Minichat/Minichat.h"
13 #include "../Graphics/Selector.h"
14 #include "../Graphics/GUI/PlayerInventoryGUI.h"
15 #include "../Graphics/GUI/PlayerInfoGUI.h"
16 #include "../Texture/PlayerEquipment.h"
17 #include "../Sound/SoundPlayer.h"
18 #include "EntityData.h"
19 #include "CitizenData.h"
20
21 class GameGUI {
22 private:
23     Window screen;
24     SDL_Rect camera{0, 0, DEFAULT_MAP_WIDTH, DEFAULT_MAP_HEIGHT};
25     TextureRepository repo;
26     SoundPlayer soundPlayer;
27     Map map;
28     Minichat minichat;
29     Selector selector;
30     PlayerInfoGUI infoGUI;
31     PlayerInventoryGUI inventoryGUI;
32     Texture& background;//Esto capaz es mejor ponerlo en window u otra clase
33
34 public:
35     GameGUI();
36
37     /* Setea el tamaño del mapa */
38     void setMapSize(int rows, int columns);
39
40     /* Carga la informacion del tile en "position" */
41     void loadTileData(Coordinate position, FloorTypeTexture floor, TextureID str
ucture,
42         CitizenData& citizen);
43
44     /* Ubica un item en el tile en "position" */
45     void createItem(Coordinate position, TextureID itemTexture);
46
47     /* Agrega un NPC al mapa*/
48     void addNPC(EntityData& entityData);
49
50     /* Agrega un NPC al mapa*/
51     void addPlayer(MapPlayerData& playerData);
52
53     /* Setea la camara en el jugador controlado por el usuario */
54     void setCameraOn(std::string& playerNickname);
55
56     /* Renderiza toda la interfaz grafica */
57     void render();
58
59     /* Actualiza el estado del mapa */
60     void update(double timeStep);
61
62     Window& getWindow();
63     Minichat& getMinichat();
64     Selector& getSelector();
65     PlayerInfoGUI& getPlayerInfo();

```

jul 21, 20 15:47

GameGUI.h

Page 2/2

```

66     PlayerInventoryGUI& getPlayerInventory();
67     Map& getMap();
68     SoundPlayer& getSoundPlayer();
69     TextureRepository &getTextureRepo();
70 };
71
72
73 #endif //ARGENTUM_GAMEGUI_H

```

jul 21, 20 15:47

GameGUI.cpp

Page 1/2

```

1  //
2  // Created by marcos on 6/25/20.
3  //
4
5  #include "GameGUI.h"
6
7  void GameGUI::setMapSize(int rows, int columns) {
8      map.setSize(rows, columns);
9  }
10
11 GameGUI::GameGUI() : repo(screen.getRenderer()), map(repo, camera, soundPlayer),
12                    minichat(screen.getRenderer()), infoGUI(screen.getRenderer())
13                    , soundPlayer)
14                    , inventoryGUI(repo, screen.getRenderer(), infoGUI),
15                    background(repo.getTexture(Background)) {
16
17 void GameGUI::loadTileData(Coordinate position, FloorTypeTexture floor,
18                             TextureID structure, CitizenData& citizen) {
19     map.loadTileData(position, floor, structure, citizen);
20 }
21
22 void GameGUI::update(double timeStep) {
23     map.update(timeStep);
24 }
25
26 void GameGUI::render() {
27     screen.clear();
28     screen.setViewport(ScreenViewport);
29     background.render(0, 0);
30
31     //Mapa
32     screen.setViewport(MapViewport);
33     map.render();
34
35     //Inventario
36     screen.setViewport(InventoryViewport);
37     inventoryGUI.render(selector.getInventorySlot());
38
39     //PlayerStats
40     screen.setViewport(PlayerInfoViewport);
41     infoGUI.render();
42
43     //Minichat
44     screen.setViewport(MinichatViewport);
45     minichat.render();
46     screen.show();
47 }
48
49 Window& GameGUI::getWindow() {
50     return screen;
51 }
52
53 Minichat &GameGUI::getMinichat() {
54     return minichat;
55 }
56
57 Selector &GameGUI::getSelector() {
58     return selector;
59 }
60
61 PlayerInfoGUI &GameGUI::getPlayerInfo() {
62     return infoGUI;
63 }
64
65 void GameGUI::addNPC(EntityData& entityData) {

```

jul 21, 20 15:47

GameGUI.cpp

Page 2/2

```

66     map.addNPC(entityData);
67 }
68
69 void GameGUI::createItem(Coordinate position, TextureID itemTexture) {
70     map.createItem(position, itemTexture);
71 }
72
73 void GameGUI::addPlayer(MapPlayerData& playerData) {
74     map.addPlayer(playerData);
75 }
76
77 PlayerInventoryGUI &GameGUI::getPlayerInventory() {
78     return inventoryGUI;
79 }
80
81 Map &GameGUI::getMap() {
82     return map;
83 }
84
85 SoundPlayer &GameGUI::getSoundPlayer() {
86     return soundPlayer;
87 }
88
89 TextureRepository& GameGUI::getTextureRepo() {
90     return repo;
91 }
92
93 void GameGUI::setCameraOn(std::string& playerNickname) {
94     map.setCameraOn(playerNickname);
95 }
96
97
98

```

jul 21, 20 15:47

GameConstants.h

Page 1/1

```

1  //
2  // Created by marcos on 6/7/20.
3  //
4
5  #ifndef ARGENTUM_GAMECONSTANTS_H
6  #define ARGENTUM_GAMECONSTANTS_H
7
8  //Dimension de los tiles
9  const int TILE_WIDTH = 128;
10 const int TILE_HEIGHT = 128;
11
12 const int TOTAL_HORIZONTAL_TILES = 100;
13 const int TOTAL_VERTICAL_TILES = 100;
14
15 const int VISIBLE_HORIZONTAL_TILES = 8;
16 const int VISIBLE_VERTICAL_TILES = 5;
17
18 //Map Viewport
19 const int DEFAULT_MAP_WIDTH = TILE_WIDTH*VISIBLE_HORIZONTAL_TILES;
20 const int DEFAULT_MAP_HEIGHT = TILE_HEIGHT*VISIBLE_VERTICAL_TILES;
21
22 //UI Viewport
23 const int DEFAULT_INVENTORY_WIDTH = 470;
24 const int DEFAULT_INVENTORY_HEIGHT = DEFAULT_MAP_HEIGHT + 296;
25
26 //Minichat Viewport
27 const int DEFAULT_MINICHAT_WIDTH = DEFAULT_MAP_WIDTH + 5;
28 const int DEFAULT_MINICHAT_HEIGHT = 210;
29
30 //PlayerStats Viewport
31 const int DEFAULT_PLAYER_INFO_WIDTH = DEFAULT_MAP_WIDTH;
32 const int DEFAULT_PLAYER_INFO_HEIGHT = 45;
33
34 //Dimension de la ventana
35 const int DEFAULT_SCREEN_WIDTH = DEFAULT_MAP_WIDTH + DEFAULT_INVENTORY_WIDTH;
36 const int DEFAULT_SCREEN_HEIGHT = DEFAULT_INVENTORY_HEIGHT;
37
38 //General para todas las barras
39 const int BAR_HEIGHT = 30;
40
41 //Barras individuales.
42 const int HEALTH_BAR_X_OFFSET = 25;
43 const int MANA_BAR_X_OFFSET = 385;
44 const int XP_BAR_X_OFFSET = 725;
45
46 //Dimension de lo que se muestra del mapa
47 const int LEVEL_WIDTH = TOTAL_HORIZONTAL_TILES * TILE_WIDTH;
48 const int LEVEL_HEIGHT = TOTAL_VERTICAL_TILES * TILE_HEIGHT;
49
50
51 #endif //ARGENTUM_GAMECONSTANTS_H

```

jul 21, 20 15:47

EntityData.h

Page 1/1

```

1 //
2 // Created by marcos on 7/2/20.
3 //
4
5 #ifndef ARGENTUM_ENTITYDATA_H
6 #define ARGENTUM_ENTITYDATA_H
7
8 #include <string>
9 #include "../Texture/TextureRepository.h"
10 #include "../Map/Coordinate.h"
11 #include "../libs/GameEnums.h"
12 #include "../Graphics/GUI/PlayerStats.h"
13 #include "../Texture/PlayerEquipment.h"
14
15 /*La info de un player para cargar en el mapa, esto me llega cuando se
16 * crea un nuevo player*/
17
18 struct EntityData {
19     TextureID texture;
20     std::string nickname;
21     Coordinate pos;
22     GameType::Direction currentDir;
23     int32_t distanceMoved;
24     int32_t level;
25 };
26
27 struct PlayerData {
28     PlayerStats generalInfo;
29     std::vector<std::tuple<TextureID, EquippedItems>> equippedItems;
30     std::vector<std::tuple<TextureID, int>> inventoryItems;
31     std::string minichatText;
32
33     PlayerData(PlayerData& other) noexcept {
34         this->generalInfo = other.generalInfo;
35         this->inventoryItems = std::move(other.inventoryItems);
36         this->equippedItems = std::move(other.equippedItems);
37         this->minichatText = std::move(other.minichatText);
38         other.generalInfo = {};
39         other.inventoryItems.clear();
40         other.equippedItems.clear();
41     }
42
43     PlayerData() = default;
44 };
45
46 struct MapPlayerData {
47     EntityData entityData;
48     GameType::Race race{};
49     PlayerEquipment equipment{};
50     bool isAlive{};
51 };
52
53 struct ItemData {
54     Coordinate position;
55     TextureID texture;
56 };
57
58 #endif //ARGENTUM_ENTITYDATA_H

```

jul 21, 20 15:47

ClientProtocol.h

Page 1/2

```

1 //
2 // Created by ivan on 24/6/20.
3 //
4
5 #ifndef ARGENTUM_CLIENTPROTOCOL_H
6 #define ARGENTUM_CLIENTPROTOCOL_H
7
8 #include "../Graphics/GUI/PlayerInventoryGUI.h"
9 #include "../libs/GameEnums.h"
10 #include <vector>
11 #include <msgpack.hpp>
12 #include "ProtocolEnumTranslator.h"
13 #include "../Texture/PlayerEquipment.h"
14 #include "../Map/Coordinate.h"
15 #include "../Graphics/GUI/PlayerStats.h"
16 #include "EntityData.h"
17
18 class Socket;
19
20 class ClientProtocol {
21 private:
22     Socket& socket;
23     ProtocolEnumTranslator translator;
24     msgpack::object_handle handler;
25     std::vector<char>* buffer{};
26
27 public:
28     /* Constructor */
29     explicit ClientProtocol(Socket& _socket) : socket(_socket) {}
30     /* Procesa la informacion del jugador recibida por el server y la asigna a un MapPlayerData */
31     MapPlayerData processAddPlayer(std::vector<char>* _buffer,
32                                     msgpack::type::tuple<GameType::Entity, std::string, int32_t>& entity
33                                     Data,
34                                     std::size_t& offset);
35     /* Procesa la informacion de un entity recibida por el server y la asigna a un EntityData */
36     EntityData processAddNPC(std::vector<char>* _buffer, msgpack::type::tuple<GameType::Entity,
37                                     std::string, int32_t> &entityData, size_t &offset);
38     /* Procesa la informacion de un Item recibida por el server y la asigna a un ItemData */
39     ItemData processAddItem(std::vector<char>* _buffer, std::size_t& offset);
40     /* Procesa la informacion del inventario y las stats del jugador recibida por el server y la asigna a un PlayerData */
41     PlayerData processAddPlayerData(std::vector<char>* _buffer);
42     /* Carga "loadBuffer" con "data" */
43     static void loadBytes(std::vector<char> &loadBuffer, void *data, unsigned int size);
44
45 private:
46     void _addManaData(PlayerData& data, size_t& offset);
47     void _addHealthData(PlayerData& data, size_t& offset);
48     void _addXPData(PlayerData& data, size_t& offset);
49     void _addEquippedItems(PlayerData& info, size_t &offset);
50     void _addClothing(PlayerData& info, size_t &offset, EquippedItems item);
51     void _addWeapon(PlayerData& info, size_t &offset);
52     void _fillInventory(PlayerData& info, size_t &offset);
53     void _addItem(PlayerData& info, GameType::ItemType type, int32_t id, int position);
54     void _addSkills(PlayerData& data, size_t &offset);
55     void _addPosition(PlayerData& data, size_t &offset);
56     void _addPlayerStats(PlayerData& data, size_t &offset);
57     void _addInventoryItems(PlayerData& data, size_t &offset);
58

```

jul 21, 20 15:47

ClientProtocol.h

Page 2/2

```

59     void _addMinichatText(PlayerData &data, size_t &offset);
60     void _addNickname(PlayerData &data, size_t &offset);
61     void _loadAddPlayerGeneralInfo(msgpack::type::tuple<GameType::Entity,
62         std::string, int32_t>& entityData, MapPlayerData& pData, std::size_t
        & offset);
63     void _loadAddPlayerEquipmentInfo(MapPlayerData& pData,
64         std::size_t& offset);
65 };
66
67
68 #endif // ARGENTUM_CLIENTPROTOCOL_H

```

jul 21, 20 15:47

ClientProtocol.cpp

Page 1/5

```

1  //
2  // Created by ivan on 24/6/20.
3  //
4
5  #include "ClientProtocol.h"
6  #include "GameGUI.h"
7  #include "../libs/Socket.h"
8
9  MSGPACK_ADD_ENUM(GameType::EventID)
10 MSGPACK_ADD_ENUM(GameType::Race)
11 MSGPACK_ADD_ENUM(GameType::FloorType)
12 MSGPACK_ADD_ENUM(GameType::Structure)
13 MSGPACK_ADD_ENUM(GameType::Entity)
14 MSGPACK_ADD_ENUM(GameType::Weapon)
15 MSGPACK_ADD_ENUM(GameType::Clothing)
16 MSGPACK_ADD_ENUM(GameType::Potion)
17 MSGPACK_ADD_ENUM(GameType::ItemType)
18 MSGPACK_ADD_ENUM(GameType::Class)
19 MSGPACK_ADD_ENUM(GameType::PlayerEvent)
20 MSGPACK_ADD_ENUM(GameType::Direction)
21
22
23 void ClientProtocol::loadBytes(std::vector<char>& loadBuffer, void* data, unsigned int size) {
24     for (unsigned int i = 0; i < size; ++i) {
25         loadBuffer[i] = *(reinterpret_cast<char*>(data) + i);
26     }
27 }
28
29 ItemData ClientProtocol::processAddItem(std::vector<char>* _buffer, std::size_t& offset) {
30     buffer = _buffer;
31     TextureID itemTexture = Nothing;
32     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
33     //Tupla itemData: ItemType, Item, positionI, positionJ
34     msgpack::type::tuple<GameType::ItemType, int32_t, int32_t, int32_t> itemData;
35
36     handler->convert(itemData);
37     GameType::ItemType itemType = std::get<0>(itemData); //Veo que tipo de item es
38
39     //Asigno la textura al item
40     if (itemType == GameType::ITEM_TYPE_WEAPON) {
41         itemTexture = translator.getWeaponDropTexture(
42             static_cast<GameType::Weapon>(std::get<1>(itemData)));
43     } else if (itemType == GameType::ITEM_TYPE_CLOTHING) {
44         itemTexture = translator.getClothingDropTexture(
45             static_cast<GameType::Clothing>(std::get<1>(itemData)));
46     } else if (itemType == GameType::ITEM_TYPE_POTION) {
47         itemTexture = translator.getPotionTexture(
48             static_cast<GameType::Potion>(std::get<1>(itemData)));
49     } else if (itemType == GameType::ITEM_TYPE_GOLD) {
50         itemTexture = Gold;
51     }
52     return {{std::get<2>(itemData), std::get<3>(itemData)}, itemTexture};
53 }
54
55 EntityData ClientProtocol::processAddNPC(std::vector<char>* _buffer, msgpack::type::tuple<GameType::Entity,
56     std::string, int32_t>& entityData, std::size_t& offset) {
57     buffer = _buffer;
58     EntityData npcData;
59     npcData.texture = translator.getEntityTexture(std::get<0>(entityData));
60     npcData.nickname = std::get<1>(entityData);
61 }

```

jul 21, 20 15:47

ClientProtocol.cpp

Page 2/5

```

62     npcData.level = std::get<2>(entityData);
63     //Tupla position: positionI, positionJ, direccion, distancia movida
64     msgpack::type::tuple<int32_t, int32_t, GameType::Direction, int32_t> positio
n;
65     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
66     handler->convert(position);
67     npcData.pos = {std::get<0>(position), std::get<1>(position)};
68     npcData.currentDir = static_cast<GameType::Direction>(std::get<2>(position))
;
69     npcData.distanceMoved = std::get<3>(position);
70     return npcData;
71 }
72
73 void ClientProtocol::_loadAddPlayerGeneralInfo(msgpack::type::tuple<GameType::En
tity,
74     std::string, int32_t>& entityData, MapPlayerData& pD
ata, std::size_t& offset) {
75     pData.entityData.texture = Nothing;
76     pData.entityData.nickname = std::get<1>(entityData);
77     pData.entityData.level = std::get<2>(entityData);
78     //Tupla position: positionI, positionJ, direccion, distancia movida
79     msgpack::type::tuple<int32_t, int32_t, GameType::Direction, int32_t> positio
n;
80     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
81     handler->convert(position);
82     pData.entityData.pos = {std::get<0>(position), std::get<1>(position)};
83     pData.entityData.currentDir = std::get<2>(position);
84     pData.entityData.distanceMoved = std::get<3>(position);
85     msgpack::type::tuple<GameType::Race> playerRace;
86     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
87     handler->convert(playerRace);
88     pData.race = std::get<0>(playerRace);
89     msgpack::type::tuple<bool> isAlive;
90     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
91     handler->convert(isAlive);
92     pData.isAlive = std::get<0>(isAlive);
93 }
94
95 void ClientProtocol::_loadAddPlayerEquipmentInfo(MapPlayerData& pData,
96     std::size_t& offset) {
97     msgpack::type::tuple<int32_t> item;
98     PlayerEquipment equipment{};
99     equipment.head = translator.getRaceTexture(
100         static_cast<GameType::Race>(pData.race));
101     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
102     handler->convert(item); /*Recibo en orden el helmet, armor, shield y weapon*
/
103     equipment.helmet = translator.getClothingTexture(
104         static_cast<GameType::Clothing>(std::get<0>(item)));
105     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
106     handler->convert(item);
107     equipment.body = translator.getClothingTexture(
108         static_cast<GameType::Clothing>(std::get<0>(item)));
109     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
110     handler->convert(item);
111     equipment.shield = translator.getClothingTexture(
112         static_cast<GameType::Clothing>(std::get<0>(item)));
113     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
114     handler->convert(item);
115     equipment.weapon = translator.getWeaponTexture(
116         static_cast<GameType::Weapon>(std::get<0>(item)));
117     pData.equipment = equipment;
118 }
119
120
121 MapPlayerData ClientProtocol::processAddPlayer(std::vector<char>* _buffer,

```

jul 21, 20 15:47

ClientProtocol.cpp

Page 3/5

```

122     msgpack::type::tuple<GameType::E
ntity,
123     std::string, int32_t>& entityDat
a, std::size_t& offset) {
124     buffer = _buffer;
125     MapPlayerData pData;
126     _loadAddPlayerGeneralInfo(entityData, pData, offset);
127     _loadAddPlayerEquipmentInfo(pData, offset);
128     return pData;
129 }
130
131 /* Agrega la informacion correspondiente al inventario a PlayerData*/
132 void ClientProtocol::_addInventoryItems(PlayerData& data, size_t& offset) {
133     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
134     //Tupla gold: cantidad de oro, cantidad de oro seguro
135     msgpack::type::tuple<int32_t, int32_t> gold;
136     handler->convert(gold);
137     data.generalInfo.gold = std::get<0>(gold);
138     data.generalInfo.safeGold = std::get<1>(gold);
139     //Aca recibe los items del inventario
140     _addEquippedItems(data, offset);
141     _fillInventory(data, offset);
142 }
143
144 /* Agrega los items equipados a PlayerData */
145 void ClientProtocol::_addEquippedItems(PlayerData& info, size_t& offset){
146     _addClothing(info, offset, Helmet);/*Esto carga el helmet
147     _addClothing(info, offset, Armor);/*Esto carga la armadura
148     _addClothing(info, offset, Shield);/*Esto carga el shield
149     _addWeapon(info, offset);
150 }
151
152 /* Llena el inventario con los items recibidos por el server */
153 void ClientProtocol::_fillInventory(PlayerData& info, size_t& offset){
154     for (int i = 0; i < 16; ++i) {
155         handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
156         msgpack::type::tuple<GameType::ItemType, int32_t> item;
157         handler->convert(item);
158         _addItem(info, std::get<0>(item), std::get<1>(item), i);
159     }
160 }
161
162 /* Agrega un item al inventario */
163 void ClientProtocol::_addItem(PlayerData& info, GameType::ItemType type, int32_t
id, int position) {
164     TextureID texture;
165     switch (type) {
166         case GameType::ITEM_TYPE_WEAPON:
167             texture = translator.getWeaponDropTexture(static_cast<GameType::Weap
on>(id));
168             break;
169         case GameType::ITEM_TYPE_CLOTHING:
170             texture = translator.getClothingDropTexture(static_cast<GameType::Cl
othing>(id));
171             break;
172         case GameType::ITEM_TYPE_POTION:
173             texture = translator.getPotionTexture(static_cast<GameType::Potion>(
id));
174             break;
175         case GameType::ITEM_TYPE_NONE:
176             texture = Nothing;
177             break;
178         default:
179             break;
180     }
181     info.inventoryItems.emplace_back(texture, position);

```

jul 21, 20 15:47

ClientProtocol.cpp

Page 4/5

```

182 }
183
184 /* Agrega las stats del jugador recibida por el server a PlayerData */
185 void ClientProtocol::_addPlayerStats(PlayerData& data, size_t& offset) {
186     _addXPData(data, offset);
187     _addManaData(data, offset);
188     _addHealthData(data, offset);
189     _addSkills(data, offset);
190     _addPosition(data, offset);
191     _addMinichatText(data, offset);
192     _addNickname(data, offset);
193 }
194
195 /* Agrega el nickname a PlayerData */
196 void ClientProtocol::_addNickname(PlayerData& data, size_t& offset) {
197     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
198     msgpack::type::tuple<std::string> name;
199     handler->convert(name);
200     data.generalInfo.nickname = std::get<0>(name);
201 }
202
203 /* Agrega un item de vestimenta a PlayerData */
204 void ClientProtocol::_addClothing(PlayerData& info, size_t& offset, EquippedItem
s item) {
205     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
206     msgpack::type::tuple<int32_t> equippedClothing;
207     handler->convert(equippedClothing);
208     info.equippedItems.emplace_back(translator.getClothingDropTexture
209         (static_cast<GameType::Clothing>(std::get<0>(equippedClothing))), it
em);
210 }
211
212 /* Agrega el arma equipada a PlayerData */
213 void ClientProtocol::_addWeapon(PlayerData& info, size_t& offset){
214     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
215     msgpack::type::tuple<int32_t> equippedWeapon;
216     handler->convert(equippedWeapon);
217     info.equippedItems.emplace_back(translator.getWeaponDropTexture(
218         static_cast<GameType::Weapon>(std::get<0>(equippedWeapon))), Weapon)
;
219 }
220
221 void ClientProtocol::_addXPData(PlayerData& data, size_t& offset) {
222     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
223     //Tupla xpData: xp actual, xp para siguiente nivel, nivel actual
224     msgpack::type::tuple<int32_t, int32_t, int32_t> xpData;
225     handler->convert(xpData);
226     data.generalInfo.xp = std::get<0>(xpData);
227     data.generalInfo.nextLevelXP = std::get<1>(xpData);
228     data.generalInfo.level = std::get<2>(xpData);
229 }
230
231 void ClientProtocol::_addHealthData(PlayerData& data, size_t& offset) {
232     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
233     //Tupla healthData: vida actual, vida total
234     msgpack::type::tuple<int32_t, int32_t> healthData;
235     handler->convert(healthData);
236     data.generalInfo.health = std::get<0>(healthData);
237     data.generalInfo.totalHealth = std::get<1>(healthData);
238 }
239
240 void ClientProtocol::_addManaData(PlayerData& data, size_t& offset) {
241     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
242     //Tupla manaData: mana actual, mana total
243     msgpack::type::tuple<int32_t, int32_t> manaData;
244     handler->convert(manaData);

```

jul 21, 20 15:47

ClientProtocol.cpp

Page 5/5

```

245     data.generalInfo.mana = std::get<0>(manaData);
246     data.generalInfo.totalMana = std::get<1>(manaData);
247 }
248
249 void ClientProtocol::_addSkills(PlayerData& data, size_t& offset){
250     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
251     //Tupla skills: strength, constitution, intelligence, agility
252     msgpack::type::tuple<int32_t, int32_t, int32_t, int32_t> skills;
253     handler->convert(skills);
254     data.generalInfo.strength = std::get<0>(skills);
255     data.generalInfo.constitution = std::get<1>(skills);
256     data.generalInfo.intelligence = std::get<2>(skills);
257     data.generalInfo.agility = std::get<3>(skills);
258 }
259
260 void ClientProtocol::_addPosition(PlayerData& data, size_t& offset) {
261     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
262     //Tupla pos: i, j
263     msgpack::type::tuple<int32_t, int32_t> pos;
264     handler->convert(pos);
265     data.generalInfo.position = {std::get<0>(pos), std::get<1>(pos)};
266 }
267
268 void ClientProtocol::_addMinichatText(PlayerData& data, size_t& offset){
269     handler = msgpack::unpack(buffer->data(), buffer->size(), offset);
270     msgpack::type::tuple<std::string> text;
271     handler->convert(text);
272     data.minichatText = std::get<0>(text);
273 }
274
275
276 PlayerData ClientProtocol::processAddPlayerData(std::vector<char>* _buffer) {
277     buffer = _buffer;
278     std::size_t offset = 0;
279     PlayerData data;
280     _addInventoryItems(data, offset);
281     _addPlayerStats(data, offset);
282     return data;
283 }

```


jul 21, 20 15:47

ClientEventHandler.h

Page 1/1

```

1 //
2 // Created by ivan on 26/6/20.
3 //
4
5 #ifndef ARGENTUM_CLIENTEVENTHANDLER_H
6 #define ARGENTUM_CLIENTEVENTHANDLER_H
7
8 #include <SDL.h>
9 #include "../libs/Thread.h"
10 #include "GameGUI.h"
11 #include <sstream>
12 #include "BlockingQueue.hpp"
13 #include "InputCommands/CommandVerifier.h"
14
15 class Socket;
16
17 /*Esta clase procesa los eventos del player y los envia al server acorde
18 * al protocolo correspondiente*/
19
20 class ClientEventHandler : public Thread {
21 private:
22     Socket& socket;
23     bool& quit;
24     GameGUI& game;
25     CommandVerifier cmdVerifier;
26     BlockingQueue<std::unique_ptr<SDL_Event>>& events;
27     std::stringstream msgBuffer;
28
29 public:
30     ClientEventHandler(Socket& _socket, bool& quit, GameGUI& game,
31         BlockingQueue<std::unique_ptr<SDL_Event>>& _events)
32         : socket(_socket), quit(quit), game(game), events(_event
33 s) {};
34
35     void run() override;
36
37 private:
38     void _handleKeyDown(SDL_Event& e);
39     void _sendMessage();
40     void _handleMouseButtonDown(SDL_Event &e);
41     void _processAttack(Coordinate coordinate);
42     void _processUseItem(int _inventorySlot);
43     void _processUnequipItem(GameType::EquipmentPlace _equipment);
44     void _processCommandInput();
45     void _handleKeyUp(SDL_Event& e);
46 };
47
48 #endif //ARGENTUM_CLIENTEVENTHANDLER_H

```

jul 21, 20 15:47

ClientEventHandler.cpp

Page 1/4

```

1 //
2 // Created by ivan on 26/6/20.
3 //
4
5 #include "ClientEventHandler.h"
6 #include "BlockingQueue.hpp"
7 #include <msgpack.hpp>
8 #include "../libs/Socket.h"
9 #include "ClientProtocol.h"
10 #include "InputCommands/InputCommand.h"
11
12 MSGPACK_ADD_ENUM(GameType::Direction)
13 MSGPACK_ADD_ENUM(GameType::PlayerEvent)
14
15 /* Procesa los eventos del usuario y manda dicho evento con su informacion corre
16 spondiente al server */
17 void ClientEventHandler::run() {
18     Minichat& minichat = game.getMinichat();
19
20     try {
21         while (!quit) {
22             std::unique_ptr<SDL_Event> e = events.pop();
23             if (e) {
24                 switch (e->type) {
25                     case SDL_QUIT:
26                         quit = true;
27                         break;
28                     case SDL_KEYDOWN:
29                         _handleKeyDown(*e);
30                         break;
31                     case SDL_KEYUP:
32                         _handleKeyUp(*e);
33                         break;
34                     case SDL_MOUSEBUTTONDOWN:
35                         _handleMouseDown(*e);
36                         break;
37                     case SDL_TEXTINPUT:
38                         minichat.handleTextInput(*e);
39                         break;
40                     case SDL_MOUSEWHEEL:
41                         minichat.handleMouseWheel(*e);
42                         break;
43                 }
44             }
45             if (msgBuffer.rdbuf()->in_avail() != 0) { /*Nos cargaron un mensaje*/
46                 _sendMessage();
47             }
48         } catch (std::exception& e) {
49             std::cerr << e.what() << std::endl;
50         }
51     }
52
53     /* Porcesa el evento de cuando se deja de apretar una tecla */
54     void ClientEventHandler::_handleKeyUp(SDL_Event& e) {
55         msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_STOP_MOVI
56 NG);
57         switch (e.key.keysym.sym) {
58             case SDLK_UP:
59                 msgpack::pack(msgBuffer, event);
60                 break;
61             case SDLK_DOWN:
62                 msgpack::pack(msgBuffer, event);
63                 break;
64             case SDLK_LEFT:
65                 msgpack::pack(msgBuffer, event);

```

jul 21, 20 15:47

ClientEventHandler.cpp

Page 2/4

```

65         break;
66     case SDLK_RIGHT:
67         msgpack::pack(msgBuffer, event);
68         break;
69     }
70 }
71
72 /* Procesa el evento de cuando se hace click */
73 void ClientEventHandler::_handleMouseDown(SDL_Event& e){
74     int clickX, clickY;
75     SDL_GetMouseState(&clickX, &clickY);
76     //Escala la posicion de click
77     clickX = (float)clickX * ((float)DEFAULT_SCREEN_WIDTH/(float)game.getWindow(
78 ).getWidth());
79     clickY = (float)clickY * ((float)DEFAULT_SCREEN_HEIGHT/(float)game.getWindow(
80 ).getHeight());
81
82     game.getMinichat().handleMouseDown({clickY, clickX}, game.getWindow());
83
84     game.getSelector().handleEvent({clickY, clickX},{game.getPlayerInfo().getYPos(
85 s(),
86         game.getPlayerInfo().getXPos(),game.getWindow());
87
88     if (e.button.button == SDL_BUTTON_RIGHT) {
89         if (Selector::hasSelectedTile({clickY, clickX})) {
90             _processAttack(game.getSelector().getSelectedTile());
91         }
92         if (Selector::hasSelectedSlot({clickY, clickX})) {
93             _processUseItem(game.getSelector().getInventorySlot());
94         }
95         if (Selector::hasSelectedEquipment({clickY, clickX})) {
96             _processUnequipItem(game.getSelector().getSelectedEquipment());
97         }
98     }
99 }
100
101 /* Procesa el evento de cuando se apreta una tecla */
102 void ClientEventHandler::_handleKeyDown(SDL_Event& e) {
103     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_START_MOV
104 ING);
105     msgpack::type::tuple<GameType::Direction> direction;
106     if (e.key.repeat == 0) {
107         switch (e.key.keysym.sym) {
108             case SDLK_UP:
109                 game.getSelector().resetTileSelection();
110                 direction = {GameType::DIRECTION_UP};
111                 msgpack::pack(msgBuffer, event);
112                 msgpack::pack(msgBuffer, direction);
113                 break;
114             case SDLK_DOWN:
115                 game.getSelector().resetTileSelection();
116                 direction = {GameType::DIRECTION_DOWN};
117                 msgpack::pack(msgBuffer, event);
118                 msgpack::pack(msgBuffer, direction);
119                 break;
120             case SDLK_LEFT:
121                 game.getSelector().resetTileSelection();
122                 direction = {GameType::DIRECTION_LEFT};
123                 msgpack::pack(msgBuffer, event);
124                 msgpack::pack(msgBuffer, direction);
125                 break;
126             case SDLK_RIGHT:
127                 game.getSelector().resetTileSelection();
128                 direction = {GameType::DIRECTION_RIGHT};
129                 msgpack::pack(msgBuffer, event);

```

jul 21, 20 15:47

ClientEventHandler.cpp

Page 3/4

```

126         msgpack::pack(msgBuffer, direction);
127         break;
128     case SDLK_BACKSPACE:
129         game.getMinichat().handleBackspace();
130         break;
131     case SDLK_RETURN:
132         _processCommandInput();
133         break;
134     case SDLK_TAB:
135         if (SoundPlayer::isMusicPlaying()) {
136             game.getSoundPlayer().pauseMusic();
137         } else {
138             game.getSoundPlayer().playMusic();
139         }
140         break;
141     }
142 }
143
144 /* Procesa el evento cuando apreto enter para ejecutar un comando del minichat */
145 void ClientEventHandler::_processCommandInput() {
146     std::string cmd = game.getMinichat().handleReturnKey();
147     if (cmd != "") { //Si apreto enter y no hay texto handleReturnKey me devuelve
148         esto
149         if (cmd == "/clear") {
150             game.getMinichat().clearMinichat();
151         } else {
152             std::unique_ptr<InputCommand> inputCmd;
153             inputCmd = cmdVerifier.verifyCommand(game, std::move(cmd));
154             if (inputCmd) {
155                 (*inputCmd)(msgBuffer); //Arma el mensaje y lo packea en msgBuffe
156             }
157         }
158     }
159 }
160
161 /* Arma el buffer y lo envia para el evento de desequipar un item */
162 void ClientEventHandler::_processUnequipItem(GameType::EquipmentPlace _equipment
163 ) {
164     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_UNEQUIP);
165     msgpack::type::tuple<int32_t> equipment;
166     equipment = _equipment;
167     msgpack::pack(msgBuffer, event);
168     msgpack::pack(msgBuffer, equipment);
169 }
170
171 /* Arma el buffer y lo envia para el evento de equiparse un item */
172 void ClientEventHandler::_processUseItem(int _inventorySlot) {
173     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_USE_ITEM)
174     ;
175     msgpack::type::tuple<int32_t> inventorySlot;
176     inventorySlot = _inventorySlot;
177     msgpack::pack(msgBuffer, event);
178     msgpack::pack(msgBuffer, inventorySlot);
179 }
180
181 /* Arma el buffer y lo envia para el evento de atacar a una posicion */
182 void ClientEventHandler::_processAttack(Coordinate selectedTile) {
183     msgpack::type::tuple<GameType::PlayerEvent> event(GameType::PLAYER_ATTACK);
184     msgpack::type::tuple<int32_t, int32_t> targetPosition;
185     targetPosition = {selectedTile.i, selectedTile.j};
186     msgpack::pack(msgBuffer, event);
187     msgpack::pack(msgBuffer, targetPosition);

```

jul 21, 20 15:47

ClientEventHandler.cpp

Page 4/4

```

187
188 /* Envia el mensaje con el evento y su informacion al servidor */
189 void ClientEventHandler::_sendMessage() {
190     std::string aux = msgBuffer.str();
191     uint32_t length = htonl(aux.size());
192     std::vector<char> buffer(sizeof(uint32_t));
193     ClientProtocol::loadBytes(buffer, &length, sizeof(uint32_t));
194     std::copy(aux.begin(), aux.end(), std::back_inserter(buffer));
195     socket.send(buffer.data(), buffer.size());
196     msgBuffer.str(""); /*Reseteo el stringstream*/
197     msgBuffer.clear();
198 }
199

```

jul 21, 20 15:47

CitizenData.h

Page 1/1

```

1  //
2  // Created by marcos on 9/7/20.
3  //
4
5  #ifndef ARGENTUM_CITIZENDATA_H
6  #define ARGENTUM_CITIZENDATA_H
7
8  #include "../Texture/TextureID.h"
9  #include <string>
10
11
12  /*Se usa cuando cargo la data inicial de los tiles, por si tienen un citizen
13  * (trader, banker, etc)*/
14  struct CitizenData {
15      TextureID texture;
16      std::string nickname;
17  };
18
19 #endif //ARGENTUM_CITIZENDATA_H

```

jul 21, 20 15:47

ArgentumClientSide.h

Page 1/1

```

1  #ifndef TP3TALLER_ARGENTUMCLIENTSIDE_H
2  #define TP3TALLER_ARGENTUMCLIENTSIDE_H
3
4  class ArgentumClientSide {
5  public:
6      /* Instancia al cliente y comienza su ejecucion */
7      static int run(int argc);
8  };
9
10
11 #endif //TP3TALLER_ARGENTUMCLIENTSIDE_H

```

jul 21, 20 15:47

ArgentumClientSide.cpp

Page 1/1

```

1  #include "ArgentumClientSide.h"
2  #include "ArgentumClient.h"
3  #include <iostream>
4
5  #define INVALID_ARGUMENTS_MESSAGE "Error: Invalid Arguments."
6  #define ARGUMENT_AMOUNT 1
7
8  int ArgentumClientSide::run(int argc) {
9      if (argc != ARGUMENT_AMOUNT) {
10         std::cerr << INVALID_ARGUMENTS_MESSAGE << std::endl;
11         return EXIT_FAILURE;
12     }
13     try {
14         Client client;
15         client.run();
16     } catch(std::exception& e) {
17         std::cerr << e.what() << std::endl;
18     }
19     return EXIT_SUCCESS;
20 }

```

jul 21, 20 15:47

ArgentumClient.h

Page 1/1

```

1  #ifndef TP3TALLER_CLIENT_H
2  #define TP3TALLER_CLIENT_H
3
4  /*Esta clase se encarga de manejar la logica de la conexion y comunicacion
5   * con el Server*/
6
7  #include "../libs/Socket.h"
8  #include "GameGUI.h"
9  #include <string>
10 #include "ClientEventHandler.h"
11
12 struct GameStartInfo;
13
14 class Client {
15 private:
16     Socket socket;
17
18 public:
19     Client();
20     Client(const Client&) = delete; /*Borro los constructores por copia*/
21     Client operator=(const Client&) = delete;
22
23     /* Comienza la ejecucion del cliente */
24     void run();
25     ~Client();
26
27 private:
28     void _gameLoop();
29     static void _initializeSDL();
30     static void _closeSDL();
31     static void _setCursor();
32 };
33
34 #endif //TP3TALLER_CLIENT_H

```

jul 21, 20 15:47

ArgentumClient.cpp

Page 1/3

```

1  #include <netdb.h>
2  #include "ArgentumClient.h"
3  #include "ClientProtocol.h"
4  #include <vector>
5  #include <utility>
6  #include "BlockingQueue.hpp"
7  #include <SDL_mixer.h>
8  #include "../UpdateEvents/UpdateEvent.h"
9  #include "UpdateReceiver.h"
10 #include "GameInitializer.h"
11 #include "../Screen/MainMenu.h"
12 #include "UpdateManager.h"
13
14 #define FREQUENCY 44100
15 #define CHUNKSIZE 2048
16 #define CURSOR_PATH "/var/Argentum/Assets/Images/UI/Cursor.bmp"
17
18
19 void Client::_gameLoop() {
20     bool quit = false;
21     GameGUI game;
22     Timer timer;
23     class MainMenu mainMenu(game.getTextureRepo().getTexture(MainMenu),
24                             game.getWindow());
25     Window& window = game.getWindow();
26     ClientProtocol protocol(socket);
27     GameInitializer initializer(game, socket, protocol);
28
29     /* Loop del menu principal */
30     mainMenu.menuScreen(quit, initializer, socket);
31
32     if (quit) return; //Si elegi salir del juego en el menu no tengo que hacer nada mas
33
34     initializer.initializeGame();
35
36     BlockingQueue<std::unique_ptr<SDL_Event>> sdlEvents;
37     UpdateManager updateManager;
38     ClientEventHandler eventHandler(socket, quit, game, sdlEvents);
39     UpdateReceiver updater(protocol, updateManager, socket, quit);
40     std::unique_ptr<SDL_Event> event(new SDL_Event());
41
42     /* Se lanzan los dos threads que van a manejar los eventos de input de usuario
43     * y los recibidos por el server respectivamente */
44     eventHandler();
45     updater();
46
47     timer.start();
48     game.getSoundPlayer().playMusic();
49     double timeStep;
50
51     try {
52         while (!quit) {
53             //Eventos de input del usuario
54             while(SDL_PollEvent(event.get()) != 0) {
55                 if (!window.handleEvent(*event)) {
56                     sdlEvents.push(std::move(event));
57                     event.reset(new SDL_Event());
58                 }
59             }
60
61             //Eventos recibidos por el servidor
62             int updatesAvailable = updateManager.updatesAvailable();
63             if (updatesAvailable > 0 ^ updatesAvailable < 5) {
64                 updatesAvailable = 1;

```

jul 21, 20 15:47

ArgentumClient.cpp

Page 2/3

```

65     } /*No updateo todas si hay menos de 5 ya que pierde fluidez la cama
66     ra (poca, pero notable)
67     * y considero que dado que nuestros updates son cada 16 ms unos 80
68     ms de atraso para este tipo de juego es imperceptible.
69     * Sin embargo, si el cliente se atrasa 5 o mas updates se las apli
70     co todas para que esto no sea un problema*/
71     for (int i = 0; i < updatesAvailable; ++i) {
72         auto update = updateManager.pop();
73         while (!update.empty()) {
74             auto updateEvent = update.pop();
75             (*updateEvent)(game);
76         }
77         timeStep = timer.getTime();
78         timer.start();
79         game.update(timeStep);
80     }
81     game.getSoundPlayer().playSounds();
82     game.render(); /*No hace falta dormir al cpu ya que el juego utiliza
83     VSYNC, por lo que el frame rate ya se cappea*/
84     } catch (std::exception& e) {
85         std::cerr << e.what() << " in Main Game Loop" << std::endl;
86     } catch (...) {
87         std::cerr << "Unknown Error in Main Game Loop" << std::endl;
88     }
89     quit = true;
90     socket.close();
91     sdlEvents.doneAdding();
92     eventHandler.join();
93     updater.join();
94 }
95 void Client::run() {
96     _gameLoop();
97 }
98
99 Client::Client() {
100     _initializeSDL();
101     _setCursor();
102 }
103
104 //Setea un cursor custom
105 void Client::_setCursor() {
106     SDL_Surface *surface;
107     SDL_Cursor *cursor;
108     surface = SDL_LoadBMP(CURSOR_PATH);
109     if (!surface) {
110         throw TPEException("Could not create cursor");
111     }
112     cursor = SDL_CreateColorCursor(surface, 0, 0);
113     SDL_FreeSurface(surface);
114     if (!cursor) {
115         throw TPEException("Could not create cursor");
116     }
117     SDL_SetCursor(cursor);
118 }
119
120 void Client::_initializeSDL() {
121     //Inicializa audio y video
122     if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO) < 0) {
123         throw TPEException("Graphics could not initialize! Graphics Error: %s\n", SDL_GetError());
124     } else {
125         //Setea filtrado de texturas lineal

```

jul 21, 20 15:47

ArgentumClient.cpp

Page 3/3

```

126     if(!SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "MipmapLinearNearest")) {
127         std::cerr << "Warning: Linear texture filtering not enabled!" << std::endl;
128     }
129     //Inicializa la carga de png
130     int imgFlags = IMG_INIT_PNG;
131     if(!IMG_Init(imgFlags) & imgFlags) {
132         SDL_Quit();
133         throw TPEException("SDL_image could not initialize! SDL_mage Error: %s\n", IMG_GetE
134         rror());
135     }
136     //Inicializa el audio, permite cargar la musica MP3
137     if(Mix_Init(MIX_INIT_MP3) != 0) {
138         IMG_Quit();
139         SDL_Quit();
140         throw TPEException("SDL_mixer could not initialize!"
141         " SDL_mixer Error: %s\n", Mix_GetError());
142     }
143     //Inicializa el reproductor de audio
144     if(Mix_OpenAudio(FREQUENCY, MIX_DEFAULT_FORMAT, 2, CHUNKSIZE) < 0) {
145         Mix_Quit();
146         IMG_Quit();
147         SDL_Quit();
148         throw TPEException("SDL_mixer could not initialize!"
149         " SDL_mixer Error: %s\n", Mix_GetError());
150     }
151     //Inicializa el cargado de fonts para texto
152     if(TTF_Init() == -1) {
153         Mix_Quit();
154         IMG_Quit();
155         SDL_Quit();
156         throw TPEException("SDL_ttf could not initialize! SDL_ttf Error:"
157         " %s\n", TTF_GetError());
158     }
159 }
160
161 void Client::_closeSDL() {
162     TTF_Quit();
163     Mix_CloseAudio();
164     Mix_Quit();
165     IMG_Quit();
166     SDL_QuitSubSystem(SDL_INIT_VIDEO | SDL_INIT_AUDIO);
167     SDL_Quit();
168 }
169
170 Client::~Client() {
171     _closeSDL();
172 }

```

jul 21, 20 15:47

Player.h

Page 1/1

```

1  //
2  // Created by marcos on 6/7/20.
3  //
4
5  #ifndef ARGENTUM_PLAYER_H
6  #define ARGENTUM_PLAYER_H
7
8  #include "Entity.h"
9  #include "../Texture/PlayerTexture.h"
10 #include "../Texture/NPCTexture.h"
11
12 /*Esta clase representa a los players en el mapa, incluido el propio*/
13
14 class Player : public Entity {
15 private:
16     PlayerTexture pTexture;
17     NPCTexture ghostTexture;
18     bool isAlive{true};
19
20 public:
21     Player(TextureRepository& repo, SDL_Rect& camera, float x, float y,
22           PlayerEquipment& images,
23           bool _isAlive = true,
24           std::string^ level = "",
25           const std::string& nickname
26           = "");
27     /* Renderiza al jugador con su equipamiento (o como fantasma si esta muerto) */
28     void render() override;
29
30     /* Añade la textura del item que se quiere equipar al jugador */
31     void equip(GameType::EquipmentPlace place, TextureID equipment);
32
33     /* Setea isAlive en false asi luego lo renderizo como fantasma */
34     void kill();
35
36     /* Setea isAlive en true asi luego lo renderizo normalmente */
37     void revive();
38
39     /*Aumenta el nivel que se muestra del player*/
40     void updateLevel(int level) override;
41 };
42
43
44 #endif //ARGENTUM_PLAYER_H

```

jul 21, 20 15:47

Player.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/7/20.
3  //
4
5  #include "Player.h"
6
7  Player::Player(TextureRepository& repo, SDL_Rect& camera, float x, float y,
8           PlayerEquipment& images, bool _isAlive, std::string^ level, const std::
9           string& nickname) :
10     Entity(camera, x, y), pTexture(repo, images, level, nickname),
11     ghostTexture(repo, PlayerGhost, "(" + level + ")", nickname) {
12     isAlive = _isAlive;
13 }
14
15 void Player::render() {
16     if (cameraFollows) {
17         Entity::updateCamera();
18     }
19     if (isAlive) {
20         Entity::render(pTexture);
21     } else {
22         Entity::render(ghostTexture);
23     }
24 }
25
26 void Player::equip(GameType::EquipmentPlace place, TextureID equipment) {
27     pTexture.equip(place, equipment);
28 }
29
30 void Player::kill() {
31     isAlive = false;
32 }
33
34 void Player::revive() {
35     isAlive = true;
36 }
37
38 void Player::updateLevel(int level) {
39     std::string strLevel = std::to_string(level);
40     pTexture.setLevel(strLevel);
41     ghostTexture.setLevel("(" + strLevel + ")");
42 }

```

jul 21, 20 15:47

NPC.h

Page 1/1

```

1  //
2  // Created by marcos on 6/8/20.
3  //
4
5  #ifndef ARGENTUM_NPC_H
6  #define ARGENTUM_NPC_H
7
8  #include "../Texture/NPCTexture.h"
9  #include "Entity.h"
10
11 /*Esta clase representa tanto a los monstruos como a los citizen (priest, banker
12 , etc)*/
13 class NPC : public Entity {
14 private:
15     NPCTexture npcTexture;
16
17 public:
18     NPC(TextureRepository& repo, SDL_Rect& camera, float x, float y,
19         TextureID texture, std::string^ level = "");
20
21     /*Aumenta el nivel que se muestra del player*/
22     void updateLevel(int level) override;
23
24     /*Renderiza en el mapa al NPC*/
25     void render() override;
26 };
27
28
29 #endif //ARGENTUM_NPC_H

```

jul 21, 20 15:47

NPC.cpp

Page 1/1

```

1  //
2  // Created by marcos on 6/8/20.
3  //
4
5  #include "NPC.h"
6
7  NPC::NPC(TextureRepository& repo, SDL_Rect &camera, float x, float y,
8      TextureID texture, std::string^ level) : Entity(camera, x, y),
9      npcTexture(repo, texture, std::move(level)) {}
10
11 void NPC::render() {
12     Entity::render(npcTexture);
13 }
14
15 void NPC::updateLevel(int level) {
16     npcTexture.setLevel(std::to_string(level));
17 }

```


jul 21, 20 15:47

Entity.h

Page 1/2

```

1  //
2  // Created by marcos on 6/9/20.
3  //
4
5  #ifndef ARGENTUM_ENTITY_H
6  #define ARGENTUM_ENTITY_H
7
8  #include "../Texture/EntityTexture.h"
9  #include "../libs/GameEnums.h"
10 #include <list>
11 #include <memory>
12 #include "../Miscellaneous/Spell.h"
13
14 /*Esta clase encapsula el comportamiento general de las entites (personajes
15  * del juego, sean npcs o players)*/
16
17 class Entity {
18 private:
19     SDL_Rect& camera;
20     int currentFrame;
21     GameType::Direction moveDirection, lastDirection;
22     float xPosition, width;
23     float yPosition, height;
24     float totalDistanceMoved{0};
25
26 protected:
27     bool cameraFollows{false};
28     std::weak_ptr<Spell> spell;
29
30 public:
31     Entity(SDL_Rect& camera, float x, float y);
32
33     /*Desplaza al entity en la direccion indicada la distanceTravelled indicada.
34     * Si reachedDestination es true, resetea el contador y el frame de la anima
35     cion.
36     * Este sera true cuando el entity haya terminado de desplazarse la distanci
37     a
38     * entre un tile y otro (el server me avisa)*/
39     GameType::Direction move(GameType::Direction direction, unsigned int distanc
40     eTravelled, bool reachedDestination);
41
42     /*Renderiza al la textura de entity recibida*/
43     void render(EntityTexture& eTexture);
44
45     /*Metodo abstracto, deben implementarlo los hijos*/
46     virtual void render() = 0;
47
48     /*Centra la camara en el player*/
49     void updateCamera();
50
51     /*Setea que la camara siga al player, este metodo solo se debera ejecutar
52     * en el player propio*/
53     void activateCamera();
54
55     /*Agrega un hechizo al entity para que lo siga*/
56     void addSpell(std::shared_ptr<Spell>& _spell);
57
58     /*Retorna el spell que guarda el entity, este metodo existe en el caso
59     * donde el entity sea matado por un hechizo y el hechizo deba migrar a un t
60     ile
61     * para no perder la animacion del hechizo*/
62     std::weak_ptr<Spell>& getSpell();
63
64     /*Setea la posicion interna del player (en pixeles)*/
65     void setPosition(float _xPosition, float _yPosition);
66

```

jul 21, 20 15:47

Entity.h

Page 2/2

```

63     /*Setea la direccion de renderizado del entity, se utiliza para cuando
64     * un entity ataca, para que mire a la direccion en la que ataco*/
65     void setLookDirection(GameType::Direction direction);
66
67     virtual void updateLevel(int level) = 0;
68
69     virtual ~Entity() = default;
70
71 private:
72     void _renderLastDirection(EntityTexture& eTexture);
73     void _modifyPosition(GameType::Direction direction, float distance);
74     void _updateFrame(bool reachedDestination);
75 };
76
77 #endif //ARGENTUM_ENTITY_H
78

```

jul 21, 20 15:47

Entity.cpp

Page 1/3

```

1  //
2  // Created by marcos on 6/9/20.
3  //
4
5  #include "Entity.h"
6  #include "../Client/GameConstants.h"
7  #include "../libs/SharedConstants.h"
8  #include "../Miscellaneous/CameraCollisionVerifier.h"
9
10 Entity::Entity(SDL_Rect &camera, float x, float y) : camera(camera) {
11     currentFrame = 0;
12     moveDirection = GameType::DIRECTION_STILL;
13     lastDirection = GameType::DIRECTION_STILL;
14     xPosition = x;
15     yPosition = y;
16     width = (float)TILE_WIDTH/2;
17     height = (float)TILE_HEIGHT;
18 }
19
20 void Entity::_updateFrame(bool reachedDestination) {
21     if (reachedDestination) {
22         if (totalDistanceMoved < static_cast<float>(TILE_WIDTH)) {
23             _modifyPosition(moveDirection, static_cast<float>(TILE_WIDTH) - totalDistanceMoved);
24         }
25         currentFrame = 0;
26         lastDirection = moveDirection;
27         moveDirection = GameType::DIRECTION_STILL;
28         totalDistanceMoved = 0;
29     } else {
30         for (int i = 0; i < 6; ++i) { /*6 es la cantidad de frames distintos del body*/
31             if (totalDistanceMoved < ((float)TILE_WIDTH/6 * (float)(i+1))) {
32                 currentFrame = i;
33                 break;
34             }
35         }
36     }
37 }
38
39 void Entity::render(EntityTexture& eTexture) {
40     if (CameraCollisionVerifier::isInsideCamera(camera, {(int)xPosition, (int)yPosition, (int)width, (int)height})) {
41         switch (moveDirection) {
42             case GameType::DIRECTION_UP:
43                 eTexture.renderBack((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, currentFrame);
44                 break;
45             case GameType::DIRECTION_DOWN:
46                 eTexture.renderFront((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, currentFrame);
47                 break;
48             case GameType::DIRECTION_RIGHT:
49                 eTexture.renderRight((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, currentFrame);
50                 break;
51             case GameType::DIRECTION_LEFT:
52                 eTexture.renderLeft((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, currentFrame);
53                 break;
54             case GameType::DIRECTION_STILL:
55                 _renderLastDirection(eTexture);
56         }
57     }
58     auto _spell = spell.lock();

```

jul 21, 20 15:47

Entity.cpp

Page 2/3

```

64     if (_spell) {
65         _spell->setPosition(xPosition, yPosition);
66         _spell->render();
67     }
68 }
69
70 void Entity::_renderLastDirection(EntityTexture& eTexture) {
71     switch (lastDirection) {
72         case GameType::DIRECTION_UP:
73             eTexture.renderBack((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, 0);
74             break;
75         case GameType::DIRECTION_DOWN:
76             eTexture.renderFront((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, 0);
77             break;
78         case GameType::DIRECTION_RIGHT:
79             eTexture.renderRight((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, 0);
80             break;
81         case GameType::DIRECTION_LEFT:
82             eTexture.renderLeft((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, 0);
83             break;
84         case GameType::DIRECTION_STILL:
85             eTexture.renderFront((int)(xPosition) - camera.x, (int)(yPosition) - camera.y, 0);
86     }
87 }
88
89 void Entity::updateCamera() {
90     //Centro la camara sobre el jugador
91     camera.x = ((int)xPosition + 55 / 2) - DEFAULT_MAP_WIDTH / 2;
92     camera.y = ((int)yPosition + 100 / 2) - DEFAULT_MAP_HEIGHT / 2;
93
94     //Mantengo la camara en los bordes
95     if (camera.x < 0) {
96         camera.x = 0;
97     }
98     if (camera.y < 0) {
99         camera.y = 0;
100     }
101     if (camera.x > LEVEL_WIDTH - camera.w) {
102         camera.x = LEVEL_WIDTH - camera.w;
103     }
104     if (camera.y > LEVEL_HEIGHT - camera.h) {
105         camera.y = LEVEL_HEIGHT - camera.h;
106     }
107 }
108
109 GameType::Direction Entity::move(GameType::Direction direction, unsigned int distanceTravelled, bool reachedDestination) {
110     float distanceInPixels = static_cast<float>(TILE_WIDTH) * static_cast<float>(distanceTravelled) / static_cast<float>(TILE_DISTANCE_IN_METERS);
111     GameType::Direction previousDirection = moveDirection;
112     moveDirection = direction;
113     _modifyPosition(direction, distanceInPixels);
114     totalDistanceMoved += distanceInPixels;
115     _updateFrame(reachedDestination);
116     return previousDirection;
117 }
118
119 void Entity::_modifyPosition(GameType::Direction direction, float distance) {
120     switch (direction) {

```

jul 21, 20 15:47

Entity.cpp

Page 3/3

```

128     case GameType::DIRECTION_UP:
129         yPosition -= distance;
130         break;
131     case GameType::DIRECTION_DOWN:
132         yPosition += distance;
133         break;
134     case GameType::DIRECTION_LEFT:
135         xPosition -= distance;
136         break;
137     case GameType::DIRECTION_RIGHT:
138         xPosition += distance;
139         break;
140     case GameType::DIRECTION_STILL:
141         //do nothing
142         break;
143     }
144 }
145
146 void Entity::activateCamera() {
147     cameraFollows = true;
148     camera.x = static_cast<int>(xPosition);
149     camera.y = static_cast<int>(yPosition);
150 }
151
152 void Entity::addSpell(std::shared_ptr<Spell>& _spell) {
153     spell = _spell;
154     _spell->setPosition(xPosition, yPosition);
155 }
156
157 std::weak_ptr<Spell> &Entity::getSpell() {
158     return spell;
159 }
160
161 void Entity::setPosition(float _xPosition, float _yPosition) {
162     xPosition = _xPosition;
163     yPosition = _yPosition;
164     moveDirection = GameType::DIRECTION_STILL;
165     lastDirection = moveDirection;
166     currentFrame = 0;
167     totalDistanceMoved = 0;
168 }
169
170 void Entity::setLookDirection(GameType::Direction direction) {
171     lastDirection = direction;
172 }

```

jul 21, 20 15:47

Table of Content

Page 1/5

1	Table of Contents					
2	1 tests.cpp.....	sheets	1 to	1 (1) pages	1- 1	8 lines
3	2 MapTests.h.....	sheets	1 to	1 (1) pages	2- 2	46 lines
4	3 MapTests.cpp.....	sheets	2 to	6 (5) pages	3- 11	512 lines
5	4 map_tests.cpp.....	sheets	6 to	7 (2) pages	12- 13	109 lines
6	5 ItemTests.h.....	sheets	7 to	7 (1) pages	14- 14	28 lines
7	6 ItemTests.cpp.....	sheets	8 to	9 (2) pages	15- 17	114 lines
8	7 item_tests.cpp.....	sheets	9 to	9 (1) pages	18- 18	27 lines
9	8 EntityTests.h.....	sheets	10 to	10 (1) pages	19- 19	37 lines
10	9 EntityTests.cpp.....	sheets	10 to	12 (3) pages	20- 24	311 lines
11	10 entity_tests.cpp....	sheets	13 to	13 (1) pages	25- 26	69 lines
12	11 ServerProtocol.h....	sheets	14 to	14 (1) pages	27- 28	63 lines
13	12 ServerProtocol.cpp..	sheets	15 to	15 (1) pages	29- 30	96 lines
14	13 ServerMonitor.h.....	sheets	16 to	16 (1) pages	31- 31	31 lines
15	14 ServerMonitor.cpp...	sheets	16 to	16 (1) pages	32- 32	30 lines
16	15 PlayerManager.h....	sheets	17 to	17 (1) pages	33- 33	38 lines
17	16 PlayerManager.cpp...	sheets	17 to	17 (1) pages	34- 34	35 lines
18	17 NonModifiableConstants.h	sheets	18 to	18 (1) pages	35- 35	14 lines
19	18 serverMain.cpp.....	sheets	18 to	18 (1) pages	36- 36	19 lines
20	19 ClientsMonitor.h....	sheets	19 to	19 (1) pages	37- 38	81 lines
21	20 ClientsMonitor.cpp..	sheets	20 to	20 (1) pages	39- 40	81 lines
22	21 ClientHandler.h.....	sheets	21 to	21 (1) pages	41- 42	91 lines
23	22 ClientHandler.cpp...	sheets	22 to	23 (2) pages	43- 46	235 lines
24	23 ClientAcceptor.h....	sheets	24 to	24 (1) pages	47- 47	53 lines
25	24 ClientAcceptor.cpp..	sheets	24 to	25 (2) pages	48- 49	105 lines
26	25 ArgentumServerSide.h	sheets	25 to	25 (1) pages	50- 50	15 lines
27	26 ArgentumServerSide.cpp	sheets	26 to	26 (1) pages	51- 51	30 lines
28	27 ArgentumServer.h....	sheets	26 to	26 (1) pages	52- 52	41 lines
29	28 ArgentumServer.cpp..	sheets	27 to	27 (1) pages	53- 54	96 lines
30	29 SaveFileManager.h...	sheets	28 to	28 (1) pages	55- 55	39 lines
31	30 SaveFileManager.cpp.	sheets	28 to	28 (1) pages	56- 56	29 lines
32	31 PlayerSaveFile.h....	sheets	29 to	29 (1) pages	57- 57	54 lines
33	32 PlayerSaveFile.cpp..	sheets	29 to	31 (3) pages	58- 61	183 lines
34	33 PlayerIndexFile.h...	sheets	31 to	31 (1) pages	62- 62	43 lines
35	34 PlayerIndexFile.cpp.	sheets	32 to	32 (1) pages	63- 64	83 lines
36	35 PlayerFilePosition.h	sheets	33 to	33 (1) pages	65- 65	17 lines
37	36 Tile.h.....	sheets	33 to	34 (2) pages	66- 67	107 lines
38	37 Tile.cpp.....	sheets	34 to	35 (2) pages	68- 70	177 lines
39	38 PointAndDistance.h..	sheets	36 to	36 (1) pages	71- 71	19 lines
40	39 Map.h.....	sheets	36 to	37 (2) pages	72- 74	150 lines
41	40 Map.cpp.....	sheets	38 to	41 (4) pages	75- 81	398 lines
42	41 InverseCoordinateDistance.h	sheets	41 to	41 (1) pages	82- 82	20 lines
43	42 InverseCoordinateDistance.cpp	sheets	42 to	42 (1) pages	83- 83	11 lines
44	43 Coordinate.h.....	sheets	42 to	42 (1) pages	84- 84	45 lines
45	44 UseReturnData.h....	sheets	43 to	43 (1) pages	85- 85	20 lines
46	45 Potion.h.....	sheets	43 to	43 (1) pages	86- 86	37 lines
47	46 Potion.cpp.....	sheets	44 to	44 (1) pages	87- 87	18 lines
48	47 ManaPotion.h.....	sheets	44 to	44 (1) pages	88- 88	24 lines
49	48 ManaPotion.cpp.....	sheets	45 to	45 (1) pages	89- 89	13 lines
50	49 HealthPotion.h.....	sheets	45 to	45 (1) pages	90- 90	23 lines
51	50 HealthPotion.cpp....	sheets	46 to	46 (1) pages	91- 91	13 lines
52	51 Gold.h.....	sheets	46 to	46 (1) pages	92- 92	39 lines
53	52 Gold.cpp.....	sheets	47 to	47 (1) pages	93- 93	24 lines
54	53 ItemsFactory.h.....	sheets	47 to	48 (2) pages	94- 95	80 lines
55	54 ItemsFactory.cpp....	sheets	48 to	48 (1) pages	95- 95	1 lines
56	55 Item.h.....	sheets	49 to	49 (1) pages	96- 97	73 lines
57	56 ItemData.h.....	sheets	50 to	50 (1) pages	98- 98	22 lines
58	57 Item.cpp.....	sheets	50 to	50 (1) pages	99- 99	52 lines
59	58 Inventory.h.....	sheets	51 to	51 (1) pages	100-101	116 lines
60	59 Inventory.cpp.....	sheets	51 to	51 (1) pages	101-101	1 lines
61	60 Shield.h.....	sheets	52 to	52 (1) pages	102-102	34 lines
62	61 Shield.cpp.....	sheets	52 to	52 (1) pages	103-103	14 lines
63	62 Head.h.....	sheets	53 to	53 (1) pages	104-104	35 lines
64	63 Head.cpp.....	sheets	53 to	53 (1) pages	105-105	14 lines
65	64 Clothing.h.....	sheets	54 to	54 (1) pages	106-106	37 lines
66	65 Clothing.cpp.....	sheets	54 to	54 (1) pages	107-107	22 lines

jul 21, 20 15:47	Table of Content	Page 2/5
66	<i>Chest.h</i> sheets 55 to 55 (1) pages 108-108 34 lines	
67	<i>Chest.cpp</i> sheets 55 to 55 (1) pages 109-109 14 lines	
68	<i>Weapon.h</i> sheets 56 to 56 (1) pages 110-110 56 lines	
69	<i>Weapon.cpp</i> sheets 56 to 56 (1) pages 111-111 57 lines	
70	<i>ShouldPlayerBeRevived.h</i> sheets 57 to 57 (1) pages 112-112 36 lines	
71	<i>ShouldPlayerBeRevived.cpp</i> sheets 57 to 57 (1) pages 113-113 57 lines	
72	<i>ShouldMonsterBeRemoved.h</i> sheets 58 to 58 (1) pages 114-114 30 lines	
73	<i>ShouldMonsterBeRemoved.cpp</i> sheets 58 to 58 (1) pages 115-115 22 lines	
74	<i>ResurrectData.h</i> sheets 59 to 59 (1) pages 116-116 21 lines	
75	<i>MonstersFactory.h</i> sheets 59 to 59 (1) pages 117-117 50 lines	
76	<i>MonstersFactory.cpp</i> sheets 60 to 60 (1) pages 118-118 52 lines	
77	<i>Game.h</i> sheets 60 to 61 (2) pages 119-121 162 lines	
78	<i>Game.cpp</i> sheets 62 to 64 (3) pages 122-126 287 lines	
79	<i>Withdraw.h</i> sheets 64 to 64 (1) pages 127-127 32 lines	
80	<i>Withdraw.cpp</i> sheets 65 to 65 (1) pages 128-128 16 lines	
81	<i>UseItem.h</i> sheets 65 to 65 (1) pages 129-129 30 lines	
82	<i>UseItem.cpp</i> sheets 66 to 66 (1) pages 130-130 29 lines	
83	<i>Unequip.h</i> sheets 66 to 66 (1) pages 131-131 32 lines	
84	<i>Unequip.cpp</i> sheets 67 to 67 (1) pages 132-132 42 lines	
85	<i>Sell.h</i> sheets 67 to 67 (1) pages 133-133 32 lines	
86	<i>Sell.cpp</i> sheets 68 to 68 (1) pages 134-134 16 lines	
87	<i>RestoreStats.h</i> sheets 68 to 68 (1) pages 135-135 32 lines	
88	<i>RestoreStats.cpp</i> sheets 69 to 69 (1) pages 136-136 17 lines	
89	<i>RequestResurrect.h</i> sheets 69 to 69 (1) pages 137-137 35 lines	
90	<i>RequestResurrect.cpp</i> sheets 70 to 70 (1) pages 138-138 30 lines	
91	<i>PlayerLeveledUp.h</i> sheets 70 to 70 (1) pages 139-139 26 lines	
92	<i>PlayerLeveledUp.cpp</i> sheets 71 to 71 (1) pages 140-140 26 lines	
93	<i>PickUpItem.h</i> sheets 71 to 71 (1) pages 141-141 35 lines	
94	<i>PickUpItem.cpp</i> sheets 72 to 72 (1) pages 142-142 34 lines	
95	<i>NotifyDeath.h</i> sheets 72 to 72 (1) pages 143-143 35 lines	
96	<i>NotifyDeath.cpp</i> sheets 73 to 73 (1) pages 144-144 50 lines	
97	<i>Move.h</i> sheets 73 to 73 (1) pages 145-145 35 lines	
98	<i>Moved.h</i> sheets 74 to 74 (1) pages 146-146 35 lines	
99	<i>Moved.cpp</i> sheets 74 to 74 (1) pages 147-147 30 lines	
100	<i>Move.cpp</i> sheets 75 to 75 (1) pages 148-148 17 lines	
101	<i>ModifyPlayerMovement.h</i> sheets 75 to 75 (1) pages 149-149 37 lines	
102	<i>ModifyPlayerMovement.cpp</i> sheets 76 to 76 (1) pages 150-150 25 lines	
103	<i>Message.h</i> sheets 76 to 76 (1) pages 151-151 34 lines	
104	<i>Message.cpp</i> sheets 77 to 77 (1) pages 152-152 19 lines	
105	<i>List.h</i> sheets 77 to 77 (1) pages 153-153 32 lines	
106	<i>List.cpp</i> sheets 78 to 78 (1) pages 154-154 15 lines	
107	<i>GetInventoryNames.h</i> sheets 78 to 78 (1) pages 155-155 30 lines	
108	<i>GetInventoryNames.cpp</i> sheets 79 to 79 (1) pages 156-156 16 lines	
109	<i>Event.h</i> sheets 79 to 79 (1) pages 157-157 22 lines	
110	<i>Event.cpp</i> sheets 80 to 80 (1) pages 158-158 6 lines	
111	<i>Drop.h</i> sheets 80 to 80 (1) pages 159-159 51 lines	
112	<i>Drop.cpp</i> sheets 81 to 81 (1) pages 160-160 52 lines	
113	<i>Drop.h</i> sheets 81 to 81 (1) pages 161-161 34 lines	
114	<i>Deposit.h</i> sheets 82 to 82 (1) pages 162-162 16 lines	
115	<i>Deposit.cpp</i> sheets 82 to 82 (1) pages 163-163 33 lines	
116	<i>Buy.h</i> sheets 83 to 83 (1) pages 164-164 16 lines	
117	<i>Buy.cpp</i> sheets 83 to 83 (1) pages 165-165 35 lines	
118	<i>Attack.h</i> sheets 84 to 84 (1) pages 166-166 48 lines	
119	<i>Attack.cpp</i> sheets 84 to 84 (1) pages 167-167 16 lines	
120	<i>UnavailablePlayerException.h</i> sheets 85 to 85 (1) pages 168-168 16 lines	
121	<i>InexistentPlayerException.h</i> sheets 85 to 85 (1) pages 169-170 126 lines	
122	<i>PlayerStats.h</i> sheets 86 to 86 (3) pages 171-175 250 lines	
123	<i>PlayerStats.cpp</i> sheets 89 to 89 (1) pages 176-177 98 lines	
124	<i>PlayerProxy.h</i> sheets 90 to 91 (2) pages 178-180 175 lines	
125	<i>PlayerProxy.cpp</i> sheets 91 to 92 (2) pages 181-183 168 lines	
126	<i>Player.h</i> sheets 93 to 95 (3) pages 184-189 363 lines	
127	<i>Player.cpp</i> sheets 96 to 96 (1) pages 190-190 16 lines	
128	<i>MovementBackup.h</i> sheets 96 to 96 (1) pages 191-191 61 lines	
129	<i>MonsterStats.h</i> sheets 97 to 97 (1) pages 192-192 65 lines	
130	<i>MonsterStats.cpp</i> sheets 97 to 98 (2) pages 193-194 75 lines	
131	<i>Monster.h</i> sheets 98 to 99 (2) pages 195-197 166 lines	
132	<i>Monster.cpp</i> sheets 98 to 99 (2) pages 195-197 166 lines	

jul 21, 20 15:47	Table of Content	Page 3/5
133	<i>Minichat.h</i> sheets 100 to 100 (1) pages 198-198 28 lines	
134	<i>Minichat.cpp</i> sheets 100 to 100 (1) pages 199-199 22 lines	
135	<i>Entity.h</i> sheets 101 to 102 (2) pages 200-202 131 lines	
136	<i>Entity.cpp</i> sheets 102 to 104 (3) pages 203-206 196 lines	
137	<i>Trader.h</i> sheets 104 to 104 (1) pages 207-207 35 lines	
138	<i>Trader.cpp</i> sheets 105 to 105 (1) pages 208-209 59 lines	
139	<i>Storage.h</i> sheets 106 to 106 (1) pages 210-211 85 lines	
140	<i>Storage.cpp</i> sheets 107 to 108 (2) pages 212-214 129 lines	
141	<i>Shop.h</i> sheets 108 to 108 (1) pages 215-215 52 lines	
142	<i>Shop.cpp</i> sheets 109 to 109 (1) pages 216-217 102 lines	
143	<i>Priest.h</i> sheets 110 to 110 (1) pages 218-218 38 lines	
144	<i>Priest.cpp</i> sheets 110 to 110 (1) pages 219-219 52 lines	
145	<i>CitizenFactory.h</i> sheets 111 to 111 (1) pages 220-220 37 lines	
146	<i>CitizenFactory.cpp</i> sheets 111 to 111 (1) pages 221-221 37 lines	
147	<i>Banker.h</i> sheets 112 to 112 (1) pages 222-223 72 lines	
148	<i>Banker.cpp</i> sheets 113 to 114 (2) pages 224-226 177 lines	
149	<i>AttackResult.h</i> sheets 114 to 114 (1) pages 227-227 15 lines	
150	<i>MapFileReader.h</i> sheets 115 to 115 (1) pages 228-228 47 lines	
151	<i>MapFileReader.cpp</i> ... sheets 115 to 116 (2) pages 229-230 69 lines	
152	<i>Configuration.h</i> sheets 116 to 117 (2) pages 231-233 136 lines	
153	<i>Configuration.cpp</i> sheets 118 to 119 (2) pages 234-236 154 lines	
154	<i>ConfigFileReader.h</i> .. sheets 119 to 120 (2) pages 237-239 166 lines	
155	<i>ConfigFileReader.cpp</i> sheets 121 to 122 (2) pages 240-243 222 lines	
156	<i>Calculator.h</i> sheets 123 to 123 (1) pages 244-245 65 lines	
157	<i>Calculator.cpp</i> sheets 124 to 124 (1) pages 246-247 122 lines	
158	<i>TPEException.h</i> sheets 125 to 125 (1) pages 248-248 21 lines	
159	<i>TPEException.cpp</i> sheets 125 to 125 (1) pages 249-249 22 lines	
160	<i>Timer.h</i> sheets 126 to 126 (1) pages 250-250 25 lines	
161	<i>Timer.cpp</i> sheets 126 to 126 (1) pages 251-251 17 lines	
162	<i>Thread.h</i> sheets 127 to 127 (1) pages 252-252 32 lines	
163	<i>Thread.cpp</i> sheets 127 to 127 (1) pages 253-253 14 lines	
164	<i>Socket.h</i> sheets 128 to 128 (1) pages 254-254 49 lines	
165	<i>Socket.cpp</i> sheets 128 to 129 (2) pages 255-256 132 lines	
166	<i>SharedConstants.h</i> ... sheets 129 to 129 (1) pages 257-257 11 lines	
167	<i>GameEnums.h</i> sheets 130 to 130 (1) pages 258-259 84 lines	
168	<i>UpdateTeleportEntity.h</i> sheets 131 to 131 (1) pages 260-260 26 lines	
169	<i>UpdateTeleportEntity.cpp</i> sheets 131 to 131 (1) pages 261-261 12 lines	
170	<i>UpdateRemoveEntity.h</i> sheets 132 to 132 (1) pages 262-262 22 lines	
171	<i>UpdateRemoveEntity.cpp</i> sheets 132 to 132 (1) pages 263-263 11 lines	
172	<i>UpdatePlayerResurrect.h</i> sheets 133 to 133 (1) pages 264-264 22 lines	
173	<i>UpdatePlayerResurrect.cpp</i> sheets 133 to 133 (1) pages 265-265 11 lines	
174	<i>UpdatePlayerDeath.h</i> sheets 134 to 134 (1) pages 266-266 22 lines	
175	<i>UpdatePlayerDeath.cpp</i> sheets 134 to 134 (1) pages 267-267 11 lines	
176	<i>UpdateMove.h</i> sheets 135 to 135 (1) pages 268-268 30 lines	
177	<i>UpdateMove.cpp</i> sheets 135 to 135 (1) pages 269-269 14 lines	
178	<i>UpdateLevelUp.h</i> sheets 136 to 136 (1) pages 270-270 24 lines	
179	<i>UpdateLevelUp.cpp</i> ... sheets 136 to 136 (1) pages 271-271 11 lines	
180	<i>UpdateGUI.h</i> sheets 137 to 137 (1) pages 272-272 22 lines	
181	<i>UpdateGUI.cpp</i> sheets 137 to 137 (1) pages 273-273 20 lines	
182	<i>UpdateEvent.h</i> sheets 138 to 138 (1) pages 274-274 20 lines	
183	<i>UpdateEquip.h</i> sheets 138 to 138 (1) pages 275-275 30 lines	
184	<i>UpdateEquip.cpp</i> sheets 139 to 139 (1) pages 276-276 32 lines	
185	<i>UpdateDestroyItem.h</i> sheets 139 to 139 (1) pages 277-277 22 lines	
186	<i>UpdateDestroyItem.cpp</i> sheets 140 to 140 (1) pages 278-278 11 lines	
187	<i>UpdateCreatePlayer.h</i> sheets 140 to 140 (1) pages 279-279 23 lines	
188	<i>UpdateCreatePlayer.cpp</i> sheets 141 to 141 (1) pages 280-280 11 lines	
189	<i>UpdateCreateNPC.h</i> ... sheets 141 to 141 (1) pages 281-281 20 lines	
190	<i>UpdateCreateNPC.cpp</i> sheets 142 to 142 (1) pages 282-282 11 lines	
191	<i>UpdateCreateItem.h</i> ... sheets 142 to 142 (1) pages 283-283 25 lines	
192	<i>UpdateCreateItem.cpp</i> sheets 143 to 143 (1) pages 284-284 38 lines	
193	<i>UpdateAttack.h</i> sheets 143 to 143 (1) pages 285-285 31 lines	
194	<i>UpdateAttack.cpp</i> ... sheets 144 to 144 (1) pages 286-287 84 lines	
195	<i>TextureRepository.h</i> sheets 145 to 145 (1) pages 288-289 64 lines	
196	<i>TextureRepository.cpp</i> sheets 146 to 149 (4) pages 290-296 433 lines	
197	<i>TextureID.h</i> sheets 149 to 150 (2) pages 297-298 99 lines	
198	<i>Texture.h</i> sheets 150 to 151 (2) pages 299-300 78 lines	

jul 21, 20 15:47	Table of Content	Page 4/5
199	198 Texture.cpp..... sheets	151 to 152 (2) pages 301-303 139 lines
200	199 PlayerTexture.h..... sheets	153 to 153 (1) pages 304-304 46 lines
201	200 PlayerTexture.cpp... sheets	153 to 154 (2) pages 305-306 100 lines
202	201 PlayerEquipment.h... sheets	154 to 154 (1) pages 307-307 21 lines
203	202 NPCTexture.h..... sheets	155 to 155 (1) pages 308-308 35 lines
204	203 NPCTexture.cpp..... sheets	155 to 155 (1) pages 309-309 56 lines
205	204 EntityTexture.h..... sheets	156 to 156 (1) pages 310-310 25 lines
206	205 EntityTexture.cpp.... sheets	156 to 156 (1) pages 311-311 12 lines
207	206 SoundRepository.h... sheets	157 to 157 (1) pages 312-312 37 lines
208	207 SoundRepository.cpp.. sheets	157 to 158 (2) pages 313-314 77 lines
209	208 SoundPlayer.h..... sheets	158 to 158 (1) pages 315-315 44 lines
210	209 SoundPlayer.cpp.... sheets	159 to 159 (1) pages 316-317 72 lines
211	210 Sound.h..... sheets	160 to 160 (1) pages 318-318 30 lines
212	211 Sound.cpp..... sheets	160 to 160 (1) pages 319-319 24 lines
213	212 Window.h..... sheets	161 to 161 (1) pages 320-320 64 lines
214	213 Window.cpp..... sheets	161 to 162 (2) pages 321-323 152 lines
215	214 MainMenu.h..... sheets	163 to 163 (1) pages 324-325 82 lines
216	215 MainMenu.cpp..... sheets	164 to 168 (5) pages 326-335 599 lines
217	216 Spell.h..... sheets	169 to 169 (1) pages 336-336 38 lines
218	217 Spell.cpp..... sheets	169 to 169 (1) pages 337-337 60 lines
219	218 CameraCollisionVerifier.h sheets	170 to 170 (1) pages 338-338 18 lines
220	219 CameraCollisionVerifier.cpp sheets	170 to 170 (1) pages 339-339 64 lines
221	220 Arrow.h..... sheets	171 to 171 (1) pages 340-340 41 lines
222	221 Arrow.cpp..... sheets	171 to 171 (1) pages 341-341 52 lines
223	222 Tile.h..... sheets	172 to 172 (1) pages 342-343 66 lines
224	223 Tile.cpp..... sheets	173 to 173 (1) pages 344-345 72 lines
225	224 Structure.h..... sheets	174 to 174 (1) pages 346-346 31 lines
226	225 Structure.cpp..... sheets	174 to 174 (1) pages 347-347 31 lines
227	226 Map.h..... sheets	175 to 176 (2) pages 348-350 138 lines
228	227 Map.cpp..... sheets	176 to 179 (4) pages 351-356 340 lines
229	228 ItemDrop.h..... sheets	179 to 179 (1) pages 357-357 31 lines
230	229 ItemDrop.cpp..... sheets	180 to 180 (1) pages 358-358 26 lines
231	230 Coordinate.h..... sheets	180 to 180 (1) pages 359-359 18 lines
232	231 Text.h..... sheets	181 to 181 (1) pages 360-360 62 lines
233	232 Text.cpp..... sheets	181 to 182 (2) pages 361-362 78 lines
234	233 Font.h..... sheets	182 to 182 (1) pages 363-363 25 lines
235	234 Font.cpp..... sheets	183 to 183 (1) pages 364-364 23 lines
236	235 Selector.h..... sheets	183 to 183 (1) pages 365-365 65 lines
237	236 Selector.cpp..... sheets	184 to 185 (2) pages 366-368 128 lines
238	237 Minichat.h..... sheets	185 to 186 (2) pages 369-370 66 lines
239	238 Minichat.cpp..... sheets	186 to 187 (2) pages 371-373 150 lines
240	239 PlayerStats.h..... sheets	188 to 188 (1) pages 374-374 27 lines
241	240 PlayerInventoryGUI.h sheets	188 to 188 (1) pages 375-375 60 lines
242	241 PlayerInventoryGUI.cpp sheets	189 to 190 (2) pages 376-378 146 lines
243	242 PlayerInfoGUI.h..... sheets	190 to 191 (2) pages 379-380 80 lines
244	243 PlayerInfoGUI.cpp... sheets	191 to 193 (3) pages 381-384 204 lines
245	244 UpdateReceiver.h... sheets	193 to 193 (1) pages 385-385 54 lines
246	245 UpdateReceiver.cpp.. sheets	194 to 195 (2) pages 386-389 222 lines
247	246 UpdateManager.h..... sheets	196 to 196 (1) pages 390-390 34 lines
248	247 UpdateManager.cpp... sheets	196 to 196 (1) pages 391-391 27 lines
249	248 Update.h..... sheets	197 to 197 (1) pages 392-392 25 lines
250	249 Update.cpp..... sheets	197 to 197 (1) pages 393-393 24 lines
251	250 ProtocolEnumTranslator.h sheets	198 to 198 (1) pages 394-395 80 lines
252	251 ProtocolEnumTranslator.cpp sheets	199 to 200 (2) pages 396-399 181 lines
253	252 clientMain.cpp..... sheets	201 to 201 (1) pages 400-400 14 lines
254	253 WithdrawCommand.h... sheets	201 to 201 (1) pages 401-401 23 lines
255	254 WithdrawCommand.cpp.. sheets	202 to 202 (1) pages 402-402 17 lines
256	255 SellCommand.h..... sheets	202 to 202 (1) pages 403-403 24 lines
257	256 SellCommand.cpp..... sheets	203 to 203 (1) pages 404-404 17 lines
258	257 ResurrectCommand.h... sheets	203 to 203 (1) pages 405-405 21 lines
259	258 ResurrectCommand.cpp sheets	204 to 204 (1) pages 406-406 17 lines
260	259 RequestInventoryNamesCommand.h sheets	204 to 204 (1) pages 407-407 17 lines
261	260 RequestInventoryNamesCommand.cpp sheets	205 to 205 (1) pages 408-408 13 lines
262	261 PickupCommand.h..... sheets	205 to 205 (1) pages 409-409 18 lines

jul 21, 20 15:47	Table of Content	Page 5/5
263	262 PickupCommand.cpp... sheets	206 to 206 (1) pages 410-410 13 lines
264	263 MessageToPlayerCommand.h sheets	206 to 206 (1) pages 411-411 22 lines
265	264 MessageToPlayerCommand.cpp sheets	207 to 207 (1) pages 412-412 17 lines
266	265 MeditateCommand.h... sheets	207 to 207 (1) pages 413-413 17 lines
267	266 MeditateCommand.cpp.. sheets	208 to 208 (1) pages 414-414 14 lines
268	267 ListCommand.h..... sheets	208 to 208 (1) pages 415-415 22 lines
269	268 ListCommand.cpp..... sheets	209 to 209 (1) pages 416-416 16 lines
270	269 InputCommand.h..... sheets	209 to 209 (1) pages 417-417 21 lines
271	270 HealCommand.h..... sheets	210 to 210 (1) pages 418-418 22 lines
272	271 HealCommand.cpp..... sheets	210 to 210 (1) pages 419-419 17 lines
273	272 DropCommand.h..... sheets	211 to 211 (1) pages 420-420 21 lines
274	273 DropCommand.cpp..... sheets	211 to 211 (1) pages 421-421 16 lines
275	274 DepositCommand.h... sheets	212 to 212 (1) pages 422-422 23 lines
276	275 DepositCommand.cpp.. sheets	212 to 212 (1) pages 423-423 16 lines
277	276 CommandVerifier.h... sheets	213 to 213 (1) pages 424-424 47 lines
278	277 CommandVerifier.cpp.. sheets	213 to 215 (3) pages 425-428 245 lines
279	278 BuyCommand.h..... sheets	215 to 215 (1) pages 429-429 23 lines
280	279 BuyCommand.cpp..... sheets	216 to 216 (1) pages 430-430 16 lines
281	280 GameInitializer.h... sheets	216 to 216 (1) pages 431-431 46 lines
282	281 GameInitializer.cpp.. sheets	217 to 218 (2) pages 432-434 158 lines
283	282 GameGUI.h..... sheets	218 to 219 (2) pages 435-436 74 lines
284	283 GameGUI.cpp..... sheets	219 to 220 (2) pages 437-438 99 lines
285	284 GameConstants.h..... sheets	220 to 220 (1) pages 439-439 52 lines
286	285 EntityData.h..... sheets	221 to 221 (1) pages 440-440 59 lines
287	286 ClientProtocol.h... sheets	221 to 222 (2) pages 441-442 69 lines
288	287 ClientProtocol.cpp.. sheets	222 to 224 (3) pages 443-447 284 lines
289	288 ClientEventHandler.h sheets	225 to 225 (1) pages 448-448 49 lines
290	289 ClientEventHandler.cpp sheets	225 to 227 (3) pages 449-452 200 lines
291	290 CitizenData.h..... sheets	227 to 227 (1) pages 453-453 20 lines
292	291 ArgentumClientSide.h sheets	228 to 228 (1) pages 454-454 12 lines
293	292 ArgentumClientSide.cpp sheets	228 to 228 (1) pages 455-455 21 lines
294	293 ArgentumClient.h... sheets	229 to 229 (1) pages 456-456 36 lines
295	294 ArgentumClient.cpp.. sheets	229 to 230 (2) pages 457-459 173 lines
296	295 Player.h..... sheets	231 to 231 (1) pages 460-460 45 lines
297	296 Player.cpp..... sheets	231 to 231 (1) pages 461-461 43 lines
298	297 NPC.h..... sheets	232 to 232 (1) pages 462-462 30 lines
299	298 NPC.cpp..... sheets	232 to 232 (1) pages 463-463 18 lines
300	299 Entity.h..... sheets	233 to 233 (1) pages 464-465 79 lines
301	300 Entity.cpp..... sheets	234 to 235 (2) pages 466-468 173 lines