Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów

Space Invaders

autor	Michał Pawłowski
prowadzący	dr hab. inż., prof. Roman Starosolski
rok akademicki	2020/2021
kierunek	informatyka
rodzaj studiów	SSI
semestr	4
termin laboratorium	wtorek 13:30
sekcja	1
termin oddania sprawozdania	2021-05-13

1 Temat

Gra Space Invaders wydana w 1978 to jedna z pierwszych gier komputerowych. Pierwotnie stworzona na dedykowane automaty do gry a następnie przeniesiona na urządzenia Atari 2600. Realizowany projekt polega na implementacji tej klasycznej gry 2D w języku C++.

2 Analiza tematu

Założenia gry są następujące. Kosmici poruszają się poziomo z jednakową prędkością, gdy znajdą się przy krawędzi planszy przenoszą się poziom niżej. Istnieje jednak pewna szansa, iż pojedynczy kosmita odłączy się od grupy i zaatakuje sam. Dodatkowo kosmici w sposób losowy oddają strzały w kierunku gracza. Nad grupą atakujących kosmitów od czasu do czasu przelatuje UFO, którego zestrzelenie gwarantuje dodatkowe punkty. Gracz przesuwa poziomo działo, którym stara się zestrzelić wszystkich kosmitów. Może się on schronić się za osłony, te jednak niszczą się wraz z zablokowanymi strzałami. Trafienie kosmity oznacza jego eliminację (usunięcie z planszy). Gracz otrzymuje 10, 20 lub 30 punktów za eliminację kosmity w zależności od jego rodzaju oraz 150 punktów i dodatkowe życie za eliminację UFO. Każdy poziom oznacza nową falę przeciwników, która różni się liczbą kosmitów. Gra kończy się gdy poziom życia gracza spadnie do 0 lub gdy kosmici dotrą na wysokość gracza, czyli do dołu planszy.

Program ma być uruchamiany w trybie okienkowym oraz wykorzystywać klawiaturę jako urządzenie wejścia. Dlatego wykorzystana została biblioteka SFML, która pozwala w prosty sposób zarządzać urządzeniami I/O oraz renderowaniem obrazu.

3 Specyfikacja zewnętrzna

3.1 Instrukcja dla użytkownika

Skompilowany program został umieszczony w folderze razem z plikami niezbędnymi do poprawnego działania. Aby uruchomić program należy wykonać plik spaceinvaders.exe

Po uruchomieniu programu pojawia się plansza startowa. Gra rozpoczyna się po naciśnięciu przycisku `enter`. Podczas gry użytkownik przesuwa działo w poziomie za pomocą klawiszy `left` i `right` oraz oddaje strzały za pomocą klawisza `spacja`. Informacje o bieżącym wyniku, poziomie oraz zdrowiu gracza wyświetlane są w górze ekranu. Po zakończeniu gry na ekranie pojawia

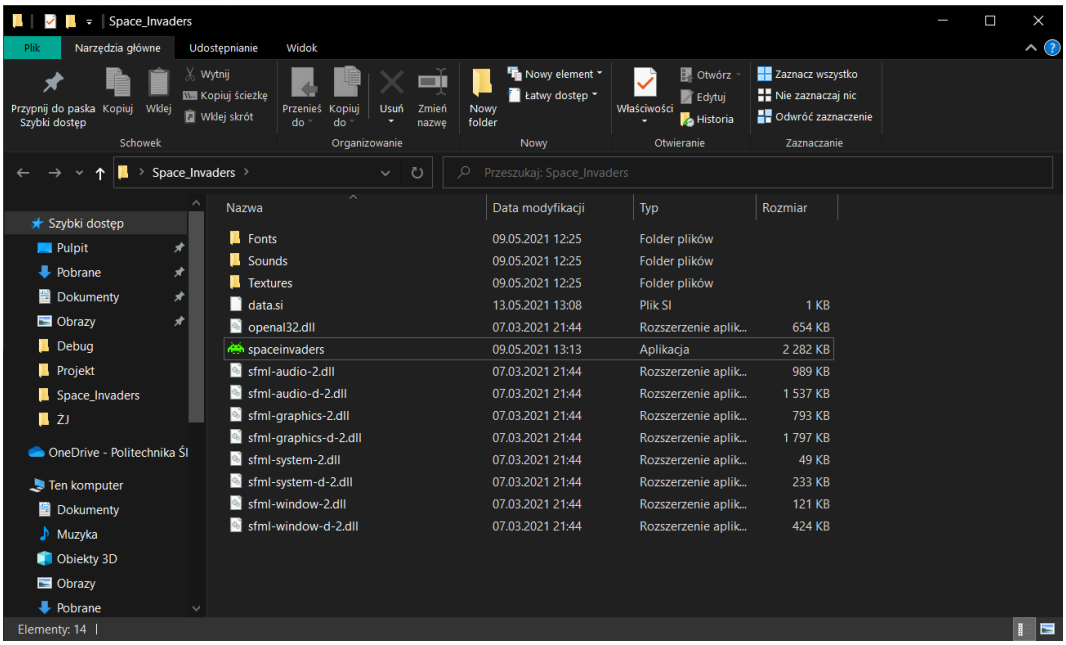
się informacyjna plansza z wynikiem uzyskanym oraz najlepszym w historii. Użytkownik może rozpocząć nową grę klawiszem `enter` lub wyjść z programu za pomocą `esc`.

Jeżeli podczas otwierania programu lub w czasie jego działania wykryty zostanie błąd, program automatycznie zakończy swoje działanie a informacja o nim zostanie wypisana na strumień błędów np.

```
ERROR[2]: Unable to open texture.
```

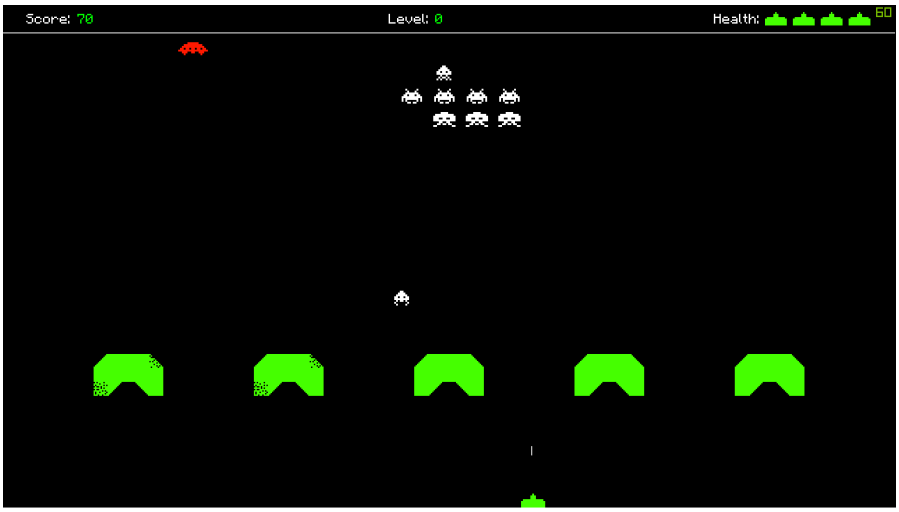
Uwaga! Do poprawnego działania programu niezbędny jest pakiet Microsoft Visual C++ 2015-2019 Redistributable (x64) w wersji 14.23.27820

3.2 Zrzuty ekranu

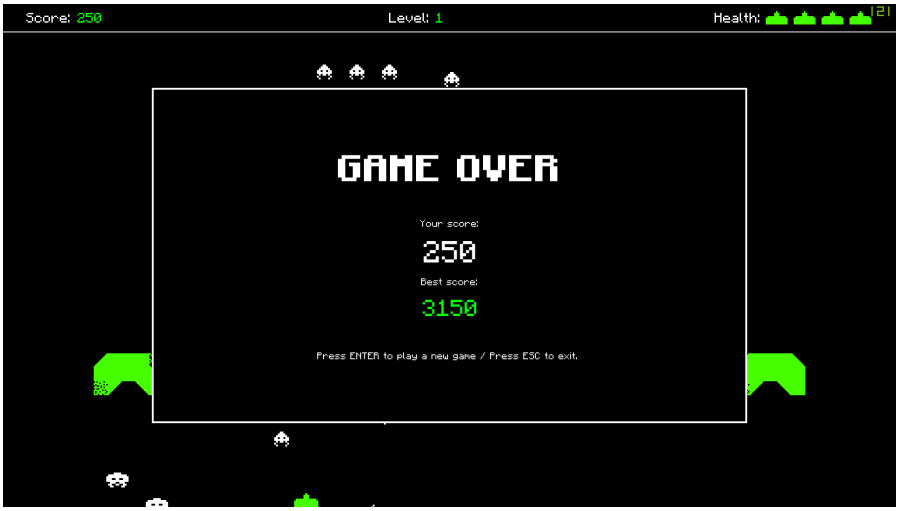


Rysunek 1: Katalog zawierający pliki gry.

Rysunek 2: Plansza startowa wyświetlana po uruchomieniu programu.



Rysunek 3: Interfejs gry.



Rysunek 4: Plansza końcowa wyświetlana po zakończeniu gry.

4 Specyfikacja wewnętrzna

4.1 Omówienie najważniejszych klas

W programie możemy wyróżnić co najmniej kilka najważniejszych klas. Silnik gry został zadeklarowany w klasie Game. Zawiera ona w sobie wszystkie obiekty, tworzone na potrzeby działania programu. Znajdziemy w niej zatem zarówno obiekty definiujące konkretne elementy gry jak i obiekty klas pochodzących z biblioteki SFML, odpowiedzialnych za otwieranie programu w oknie, renderowania itp. Metody zadeklarowane w tej klasie to zbiór metod stanowiących główną mechanikę gry.

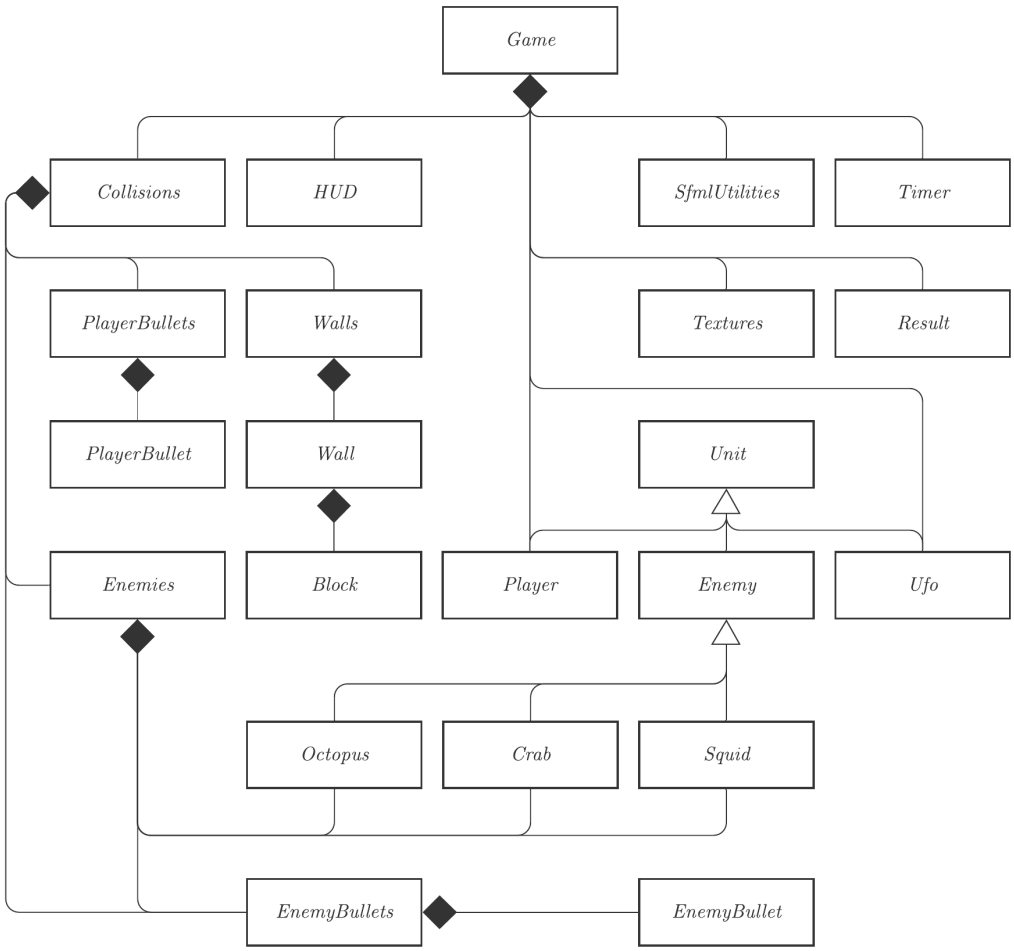
Zadaniem klasy Wall jest poprawne skonstruowanie i przechowywanie tablicy obiektów klasy Block, która definiuje pojedynczy blok osłony za którą może się chować gracz. Klasa Walls przechowuje w wektorze obiekty Wall oraz zarządza nimi.

Unit to abstrakcyjna klasa bazowa po której dziedziczą wszystkie klasy definiujące jednostki (postacie) z gry. Zawiera ona głównie pola i metody wywodzące się z biblioteki SFML. Bezpośrednio po tej klasie dziedziczy Player oraz Ufo. W pierwszej klasie zaimplementowane są metody umożliwiające interakcję z użytkownikiem, czyli sterowanie i strzelanie. Druga klasa definiuje obiekt UFO, który od czasu do czasu pojawia się na ekranie. Trzecią klasą dziedziczącą po Unit jest klasa abstrakcyjna Enemy. Stanowi ona klasę bazową dla klas Squid, Crab i Octopus. Klasy pochodne w różny sposób deklarują

metody wirtualne klasy bazowej, tak by zachowanie na planszy przeciwników innych klasy było zróżnicowane. Obiekty tych klas znajdują się w strukturze danych zawartej w klasie Enemies.

Klasa Collisions implementuje metody niezbędne do wykrywania kolizji pomiędzy wszelkimi obiektami.

4.2 Hierarchia klas



Rysunek 5: Diagram klas ULM.

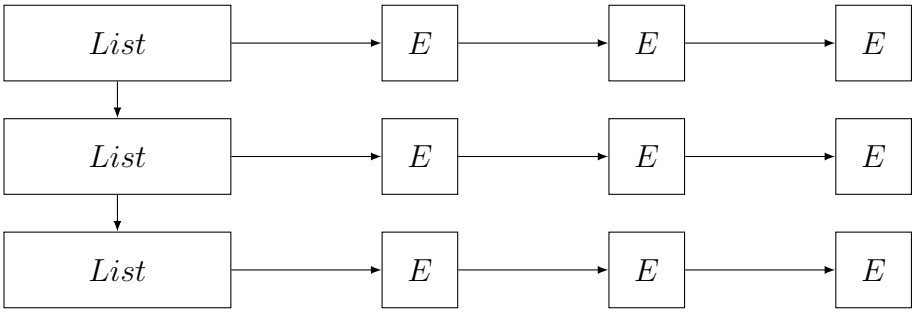
4.3 Szczegółowy opis klas i metod

Szczegółowy opis klas i metod zostanie wygenerowany przy pomocy Doxygen'a i załączony do finalnego sprawozdania.

4.4 Struktury danych

Ilość przeciwników jest zmienna w zależności od danego poziomu. Dlatego też obiekty klas pochodnych wirtualnej klasy Enemy są przechowywane w liście list znajdującej się w obiekcie klasy Enemies. Jest to największa struktura danych w programie. Oprócz niej wykorzystywane są pomniejsze struktury danych takie jak: mapy (do przechowywania wskaźników na tekstury) oraz wektory (do przechowywania obiektów klas Wall, Block, PlayerBullet , EnemyBullet.

Wszystkie struktury danych zostały stworzone w oparciu o kontenery STL typu list , vector czy map.



Rysunek 6: Przykładowy schemat listy list.

4.5 Ogólny schemat działania programu

Program rozpoczyna się od utworzenia obiektu klasy Game. W trakcie konstrukcji tworzone są również wszystkie zadeklarowane w niej obiekty. W obiekcie klasy Enemies zainicjowana zostaje struktura danych zawierająca obiekty klasy Enemy. Jej elementy stopniowo są usuwane wraz z eliminacjami dokonanymi przez gracza. Po przejściu do kolejnego poziomu struktura zostaje zainicjowana na nowo, tym razem z inną liczbą elementów. Program działa w nieskończonej pętli, dopóki okno aplikacji pozostaje otwarte, wywołując metody update() oraz render(). Pierwsza z nich odpowiedzialna jest za sczytanie sygnałów urządzeń wejściowych oraz wprowadzanie zmian w obiektach, tak by symulować postęp gry. Druga natomiast odpowiada za wy-

świetlanie zmian w oknie aplikacji. Przed zakończeniem działania programu zwolniona zostaje zaalokowana pamięć.

4.6 Biblioteki zewnętrzne i techniki obiektowe

Program opiera się o technikę programowania obiektowego. Wykorzystane zostały m.in.: klasy, klasy abstrakcyjne z metodami wirtualnymi, dziedziczenie, hermetyzacja, polimorfizm. W programie zawarte zostały również elementy języka C++ przedstawione na laboratoriach: mechanizm wyjątków, kontenery STL, algorytmy i iteratory STL oraz szablony funkcji. Program wykorzystuje bibliotekę Simple and Fast Multimedia Library (SFML).

5 Testowanie

Program był testowany zarówno na bieżąco podczas programowania jak i w finalnej wersji przez niezależne osoby. Podczas programowania wykryto typowe błędy programistyczne takie jak m.in.: wyjście po za zakres, odwołanie do usuniętego obiektu. Wykryte błędy skorygowano. Przetestowano wszystkie możliwe kolizje. Wynikające z nich usuwanie obiektów ze struktur danych, nie generuje problemów ani wycieków pamięci. Finalny program został przetestowany przez 3 niezależne osoby. Podczas testów nie wykryły one żadnych błędów.

6 Wnioski

Stworzenie prostej gry 2D daje możliwość przećwiczenia oraz wykazania się umiejętnościami programowania obiektowego. W porównaniu do projektów z poprzednich semestrów, wykorzystanie kontenerów STL zamiast struktur samodzielnie zaimplementowanych znacznie upraszcza i przyspiesza proces pisania programu. Wykorzystanie zewnętrznej biblioteki SFML pozwoliło na stworzenie programu w trybie okienkowym skupiając się na paradygmatach programowania obiektowego bez zagłębiania się w dokumentację OpenGL.

7 Link do repozytorium

https://github.com/polsl-aei-pk4/0fa12ac6-gr21-repo/tree/main/Projekt

Dodatek
Szczegółowy opis typów i funkcji

Space Invaders

Generated by Doxygen 1.8.17

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Block Class Reference

`#include <Block.h>`

Public Member Functions

- Block (sf::Texture *texture, bool isSquareParam, float xPos, float yPos, float rotation)
- const sf::FloatRect getBounds () const
- bool getIsSquare () const
- float getSpriteRotation () const
- int & getStageAccess ()
- void setNewTexture (sf::Texture *newTexture)
- void render (sf::RenderWindow *window) const

3.1.1 Detailed Description

Declaration of Walls class

Author

Michal Pawlowski

Date

2021-05-11

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Block()

```
Block::Block (
            sf::Texture * texture,
            bool isSquareParam,
            float xPos,
            float yPos,
            float rotation )
```

constructor

3.1.3 Member Function Documentation

3.1.3.1 getBounds()

```
const sf::FloatRect Block::getBounds ( ) const
```

give info about sprite coverage

3.1.3.2 getIsSquare()

```
bool Block::getIsSquare ( ) const
```

give information about shape

3.1.3.3 getSpriteRotation()

```
float Block::getSpriteRotation ( ) const
```

give information about sprite rotation

3.1.3.4 getStageAccess()

```
int & Block::getStageAccess ( )
```

give access to stage

3.1.3.5 render()

```
void Block::render (
            sf::RenderWindow * window ) const
```

draw whis object

3.1.3.6 setNewTexture()

```
void Block::setNewTexture (
            sf::Texture * newTexture )
```

set new texture

The documentation for this class was generated from the following files:

- Block.h
- Block.cpp

3.2 Collision Class Reference

```
#include <Collisions.h>
```

Public Member Functions

- Collision (sf::SoundBuffer *explsionSoundParam)
- void checkCollisions (int &score, bool &isGameEnded, Player &player, Ufo &ufo, vector< PlayerBullet > &bullets, vector< EnemyBullet > &enemyBullets, list< list< Enemy * >> &mainList, Walls &wallsParam)

3.2.1 Detailed Description

Declaration of Collisions class

Author

Michal Pawlowski

Date

2021-05-11

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Collision()

```
Collision::Collision (
            sf::SoundBuffer * explsionSoundParam )
```

constructor

3.2.3 Member Function Documentation

3.2.3.1 checkCollisions()

```
void Collision::checkCollisions (
            int & score,
            bool & isGameEnded,
            Player & player,
            Ufo & ufo,
            vector< PlayerBullet > & bullets,
            vector< EnemyBullet > & enemyBullets,
            list< list< Enemy * >> & mainList,
            Walls & wallsParam )
```

check all possible collisions

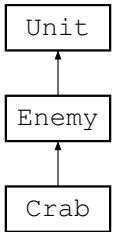
The documentation for this class was generated from the following files:

- Collisions.h
- Collisions.cpp

3.3 Crab Class Reference

```
#include <Crab.h>
```

Inheritance diagram for Crab:



Public Member Functions

- **Crab** (sf::Texture *crab1, sf::Texture *crab2, float x, float y)

Additional Inherited Members

3.3.1 Detailed Description

Declaration of Crab class

Author

Michal Pawlowski

Date

2021-05-11

The documentation for this class was generated from the following files:

- Crab.h
- Crab.cpp

3.4 Enemies Class Reference

`#include <Enemies.h>`

Public Member Functions

- Enemies (sf::Texture *octopus1, sf::Texture *octopus2, sf::Texture *crab1, sf::Texture *crab2, sf::Texture *squid1, sf::Texture *squid2, sf::Texture *enemyBullet)
- ~Enemies ()
- void update (int &level, bool &isGameEnded, PlayerBullets &playerBullets, float deltaTime)
- void render (sf::RenderWindow *window)
- list< list< Enemy * > > & getEnemyList ()
- vector< EnemyBullet > & getBulletsVector ()
- void deleteAllEnemiesAndBullets ()

3.4.1 Detailed Description

Declaration of Group class

Author

Michal Pawlowski

Date

2021-05-11

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Enemies()

```
Enemies::Enemies (
            sf::Texture * octopus1,
            sf::Texture * octopus2,
            sf::Texture * crab1,
            sf::Texture * crab2,
            sf::Texture * squid1,
            sf::Texture * squid2,
            sf::Texture * enemyBullet )
```

construcor

3.4.2.2 ~Enemies()

```
Enemies::~Enemies ( )
```

destrucor

3.4.3 Member Function Documentation

3.4.3.1 deleteAllEnemiesAndBullets()

```
void Enemies::deleteAllEnemiesAndBullets ( )
```

delete all enemies and their bullets

3.4.3.2 getBulletsVector()

```
vector< EnemyBullet > & Enemies::getBulletsVector ( )
```

give access to bullets vector

3.4.3.3 getEnemyList()

```
list< list< Enemy * > > & Enemies::getEnemyList ( )
```

give access to list of enemies

3.4.3.4 render()

```
void Enemies::render (
            sf::RenderWindow * window )
```

draws all enemies from group

3.4.3.5 update()

```
void Enemies::update (
            int & level,
            bool & isGameEnded,
            PlayerBullets & playerBullets,
            float deltaTime )
```

update events

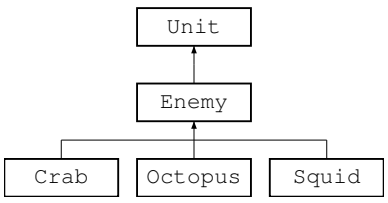
The documentation for this class was generated from the following files:

- Enemies.h
- Enemies.cpp

3.5 Enemy Class Reference

```
#include <Enemy.h>
```

Inheritance diagram for Enemy:



Public Member Functions

- Enemy ()
- const bool isAttackPossible ()
- void update (float deltaTime, bool leftDirection)
- bool checkSideBorderCollision (bool leftDirection)
- bool checkBottomBorderCollision ()
- void moveToLowerRow ()
- enemyType getEnemyType ()

Protected Attributes

- sf::Texture * texture2
- enemyType type
- bool underSoloAttack = false
- float timeToAnimate = 10.f

Additional Inherited Members

3.5.1 Detailed Description

Declaration of Enemy virtual class

Author

Michal Pawlowski

Date

2021-05-11

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Enemy()

```
Enemy::Enemy ( )
```

no argument constructor

3.5.3 Member Function Documentation

3.5.3.1 checkBottomBorderCollision()

```
bool Enemy::checkBottomBorderCollision ( )
```

checks if object collide with horizontal window borderline

3.5.3.2 checkSideBorderCollision()

```
bool Enemy::checkSideBorderCollision (
            bool leftDirection )
```

checks if object collide with vertical window borderline

3.5.3.3 getEnemyType()

```
enemyType Enemy::getEnemyType ( )
```

return type of alien

3.5.3.4 isAttackPossible()

```
const bool Enemy::isAttackPossible ( )  [virtual]
```

inform is attack possible

Implements Unit.

3.5.3.5 moveToLowerRow()

```
void Enemy::moveToLowerRow ( )
```

moves enemy to lower row

3.5.3.6 update()

```
void Enemy::update (
            float deltaTime,
            bool leftDirection )
```

update state

3.5.4 Member Data Documentation

3.5.4.1 texture2

```
sf::Texture* Enemy::texture2  [protected]
```

second texture

3.5.4.2 timeToAnimate

```
float Enemy::timeToAnimate = 10.f  [protected]
```

define time to change texture

3.5.4.3 type

```
enemyType Enemy::type  [protected]
```

define type of alien

3.5.4.4 underSoloAttack

```
bool Enemy::underSoloAttack = false  [protected]
```

define is enemy under solo attack

The documentation for this class was generated from the following files:

- Enemy.h
- Enemy.cpp

3.6 EnemyBullet Class Reference

```
#include <EnemyBullet.h>
```

Public Member Functions

- EnemyBullet (float xPosition, float yPosition, float xDirection, float yDirection, sf::Texture ∗texture)
- const sf::FloatRect getBounds () const
- void update (float deltaTime)
- void render (sf::RenderTarget ∗target) const

3.6.1 Detailed Description

Declaration of EnemyBullets class

Author

Michal Pawlowski

Date

2021-05-11

3.6.2 Constructor & Destructor Documentation

3.6.2.1 EnemyBullet()

```
EnemyBullet::EnemyBullet (
            float xPosition,
            float yPosition,
            float xDirection,
            float yDirection,
            sf::Texture * texture )
```

construcor

3.6.3 Member Function Documentation

3.6.3.1 getBounds()

```
const sf::FloatRect EnemyBullet::getBounds ( ) const
```

give info about shape coverage

3.6.3.2 render()

```
void EnemyBullet::render (
            sf::RenderTarget * target ) const
```

render bullet

3.6.3.3 update()

```
void EnemyBullet::update (
            float deltaTime )
```

update bullet sate

The documentation for this class was generated from the following files:

- EnemyBullet.h
- EnemyBullet.cpp

3.7 EnemyBullets Class Reference

```
#include <EnemyBullets.h>
```

Public Member Functions

- EnemyBullets (sf::Texture ∗texture)
- ∼EnemyBullets ()
- void updateBullets (float deltaTime)
- void spawnBullet (float xPos, float yPos)
- void renderBullets (sf::RenderWindow ∗windowPtr) const
- vector< EnemyBullet > & getBulletsVector ()
- void deleteAllEnemyBullets ()

3.7.1 Detailed Description

Declaration of EnemyBullets class

Author

Michal Pawlowski

Date

2021-05-11

3.7.2 Constructor & Destructor Documentation

3.7.2.1 EnemyBullets()

```
EnemyBullets::EnemyBullets (
            sf::Texture * texture )
```

constructor

3.7.2.2 ∼EnemyBullets()

```
EnemyBullets::∼EnemyBullets ( )
```

destructor

3.7.3 Member Function Documentation

3.7.3.1 deleteAllEnemyBullets()

```
void EnemyBullets::deleteAllEnemyBullets ( )
```

delete all enemy bullets

3.7.3.2 getBulletsVector()

```
vector< EnemyBullet > & EnemyBullets::getBulletsVector ( )
```

give access to bullets vector

3.7.3.3 renderBullets()

```
void EnemyBullets::renderBullets (
            sf::RenderWindow * windowPtr ) const
```

render bullets

3.7.3.4 spawnBullet()

```
void EnemyBullets::spawnBullet (
            float xPos,
            float yPos )
```

add new bullet

3.7.3.5 updateBullets()

```
void EnemyBullets::updateBullets (
            float deltaTime )
```

update bullets status

The documentation for this class was generated from the following files:

- EnemyBullets.h
- EnemyBullets.cpp

3.8 Game Class Reference

```
#include <Game.h>
```

Public Member Functions

- Game ()
- ~Game ()
- const bool isWindowOpened () const
- void update ()
- void render ()

3.8.1 Detailed Description

Declaration of Game engine class

Author

Michal Pawlowski

Date

2021-05-11

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Game()

```
Game::Game ( )
```

no argument constructor

3.8.2.2 ~Game()

```
Game::~Game ( )
```

destructor

3.8.3 Member Function Documentation

3.8.3.1 isWindowOpened()

```
const bool Game::isWindowOpened ( ) const
```

return information is program window open

3.8.3.2 render()

```
void Game::render ( )
```

render current state

3.8.3.3 update()

```
void Game::update ( )
```

update events

The documentation for this class was generated from the following files:

- Game.h
- Game.cpp

3.9 HUD Class Reference

```
#include <HUD.h>
```

Public Member Functions

- HUD (sf::Texture ∗texture, sf::Font ∗font)
- void updateHUD (int score, int level)
- void renderHUD (sf::RenderWindow ∗windowPtr, int playerHealth)
- void renderHealthBar (sf::RenderWindow ∗windowPtr, int playerHealth)

3.9.1 Detailed Description

Declaration of HUD class

Author

Michal Pawlowski

Date

2021-05-11

3.9.2 Constructor & Destructor Documentation

3.9.2.1 HUD()

```
HUD::HUD (
            sf::Texture * texture,
            sf::Font * font )
```

constructor

3.9.3 Member Function Documentation

3.9.3.1 renderHealthBar()

```
void HUD::renderHealthBar (
            sf::RenderWindow * windowPtr,
            int playerHealth )
```

render health bar

3.9.3.2 renderHUD()

```
void HUD::renderHUD (
            sf::RenderWindow * windowPtr,
            int playerHealth )
```

render HUD

3.9.3.3 updateHUD()

```
void HUD::updateHUD (
            int score,
            int level )
```

updates HUD information

The documentation for this class was generated from the following files:

- HUD.h
- HUD.cpp

3.10 Menu Class Reference

```
#include <Menu.h>
```

Public Member Functions

- Menu (sf::Texture ∗start, sf::Texture ∗end, sf::Font ∗font)
- void startMenu (sf::RenderWindow ∗window)
- void endMenu (sf::RenderWindow ∗window)
- void scaleText (float deltaTime)
- void updateScore (int score, int bestScore)

3.10.1 Detailed Description

Declaration of Menu class

Author

Michal Pawlowski

Date

2021-05-11

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Menu()

```
Menu::Menu (
            sf::Texture * start,
            sf::Texture * end,
            sf::Font * font )
```

constructor

3.10.3 Member Function Documentation

3.10.3.1 endMenu()

```
void Menu::endMenu (
            sf::RenderWindow * window )
```

displays end menu

3.10.3.2 scaleText()

```
void Menu::scaleText (
            float deltaTime )
```

scale menu text

3.10.3.3 startMenu()

```
void Menu::startMenu (
            sf::RenderWindow * window )
```

displays start menu

3.10.3.4 updateScore()

```
void Menu::updateScore (
            int score,
            int bestScore )
```

save score to displayed text

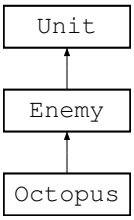
The documentation for this class was generated from the following files:

- Menu.h
- Menu.cpp

3.11 Octopus Class Reference

`#include <Octopus.h>`

Inheritance diagram for Octopus:



Public Member Functions

- **Octopus** (sf::Texture ∗crab1, sf::Texture ∗crab2, float x, float y)

Additional Inherited Members

3.11.1 Detailed Description

Declaration of Enemy class

Author

Michal Pawlowski

Date

2021-05-11

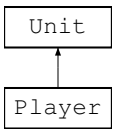
The documentation for this class was generated from the following files:

- Octopus.h
- Octopus.cpp

3.12 Player Class Reference

`#include <Player.h>`

Inheritance diagram for Player:



Public Member Functions

- Player (sf::Texture *cannonTexture, sf::Texture *destroyedCannonTexture)
- int getHealth ()
- void decreaseHealth ()
- void increaseHealth ()
- const bool isAttackPossible ()
- void update (float deltaTime)
- void setNewCannon ()

Additional Inherited Members

3.12.1 Detailed Description

Declaration of Player class

Author

Michal Pawlowski

Date

2021-05-11

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Player()

```
Player::Player (
            sf::Texture * cannonTexture,
            sf::Texture * destroyedCannonTexture )
```

no argument constructor

3.12.3 Member Function Documentation

3.12.3.1 decreaseHealth()

```
void Player::decreaseHealth ( )
```

decrease value of player health

3.12.3.2 getHealth()

```
int Player::getHealth ( )
```

return value of player health

3.12.3.3 increaseHealth()

```
void Player::increaseHealth ( )
```

decrease value of player health

3.12.3.4 isAttackPossible()

```
const bool Player::isAttackPossible ( )  [virtual]
```

inform is attack possible

Implements Unit.

3.12.3.5 setNewCannon()

```
void Player::setNewCannon ( )
```

set new cannon texture

3.12.3.6 update()

```
void Player::update (
            float deltaTime )
```

update state

The documentation for this class was generated from the following files:

- Player.h
- Player.cpp

3.13 PlayerBullet Class Reference

`#include <PlayerBullet.h>`

Public Member Functions

- PlayerBullet (float xPosition, float yPosition, float xDirection, float yDirection)
- const sf::FloatRect getBounds () const
- void update (float deltaTime)
- void render (sf::RenderTarget *target) const

3.13.1 Detailed Description

Declaration of PlayerBullet class

Author

Michal Pawlowski

Date

2021-05-11

3.13.2 Constructor & Destructor Documentation

3.13.2.1 PlayerBullet()

```
PlayerBullet::PlayerBullet (
            float xPosition,
            float yPosition,
            float xDirection,
            float yDirection )
```

construcor

3.13.3 Member Function Documentation

3.13.3.1 getBounds()

```
const sf::FloatRect PlayerBullet::getBounds ( ) const
```

give info about shape coverage

3.13.3.2 render()

```
void PlayerBullet::render (
            sf::RenderTarget * target ) const
```

render bullet

3.13.3.3 update()

```
void PlayerBullet::update (
            float deltaTime )
```

update bullet sate

The documentation for this class was generated from the following files:

- PlayerBullet.h
- PlayerBullet.cpp

3.14 PlayerBullets Class Reference

```
#include <PlayerBullets.h>
```

Public Member Functions

- PlayerBullets (sf::SoundBuffer *shootSoundParam)
- ~PlayerBullets ()
- void updateBullets (float deltaTime)
- void spawnBullet (float xPos)
- void renderBullets (sf::RenderWindow *windowPtr) const
- vector< PlayerBullet > & getBulletsVector ()
- void deleteAllBullets ()

3.14.1 Detailed Description

Declaration of PlayerBullets class

Author

Michal Pawlowski

Date

2021-05-11

3.14.2 Constructor & Destructor Documentation

3.14.2.1 PlayerBullets()

```
PlayerBullets::PlayerBullets (
            sf::SoundBuffer * shootSoundParam )
```

constructor

3.14.2.2 ~PlayerBullets()

```
PlayerBullets::~PlayerBullets ( )
```

destructor

3.14.3 Member Function Documentation

3.14.3.1 deleteAllBullets()

```
void PlayerBullets::deleteAllBullets ( )
```

delete all bullets

3.14.3.2 getBulletsVector()

```
vector< PlayerBullet > & PlayerBullets::getBulletsVector ( )
```

give access to bullets vector

3.14.3.3 renderBullets()

```
void PlayerBullets::renderBullets (
            sf::RenderWindow * windowPtr ) const
```

render bullets

3.14.3.4 spawnBullet()

```
void PlayerBullets::spawnBullet (
            float xPos )
```

add new bullet

3.14.3.5 updateBullets()

```
void PlayerBullets::updateBullets (
            float deltaTime )
```

update bullets status

The documentation for this class was generated from the following files:

- PlayerBullets.h
- PlayerBullets.cpp

3.15 Result Class Reference

```
#include <Result.h>
```

Public Member Functions

- Result ()
- ~Result ()
- int & getScore ()
- int & getBestScore ()
- int & getLevel ()
- void resetScore ()
- void resetLevel ()
- void updateBestScore ()

3.15.1 Detailed Description

Declaration of Result class

Author

Michal Pawlowski

Date

2021-05-11

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Result()

```
Result::Result ( )
```

constructor

3.15.2.2 ~Result()

```
Result::~Result ( )
```

destructor

3.15.3 Member Function Documentation

3.15.3.1 getBestScore()

```
int & Result::getBestScore ( )
```

give acces to best score

3.15.3.2 getLevel()

```
int & Result::getLevel ( )
```

give acces to level

3.15.3.3 getScore()

```
int & Result::getScore ( )
```

give acces to score

3.15.3.4 resetLevel()

```
void Result::resetLevel ( )
```

reset level

3.15.3.5 resetScore()

```
void Result::resetScore ( )
```

reset score

3.15.3.6 updateBestScore()

```
void Result::updateBestScore ( )
```

update new best score

The documentation for this class was generated from the following files:

- Result.h
- Result.cpp

3.16 SfmlUtilities Class Reference

```
#include <SfmlUtilities.h>
```

Public Member Functions

- SfmlUtilities ()
- ~SfmlUtilities ()
- void inputCheck (bool &isGameStarted, bool &isGameEnded, bool &isNewGameStarted)
- sf::RenderWindow * getWindow ()
- sf::Event & getEvent ()
- const bool isWindowOpened () const

3.16.1 Detailed Description

Declaration of SfmlUtiliteis class

Author

Michal Pawlowski

Date

2021-05-11

3.16.2 Constructor & Destructor Documentation

3.16.2.1 SfmlUtilities()

```
SfmlUtilities::SfmlUtilities ( )
```

no argument constructor

3.16.2.2 ~SfmlUtilities()

```
SfmlUtilities::~SfmlUtilities ( )
```

destructor

3.16.3 Member Function Documentation

3.16.3.1 getEvent()

```
sf::Event & SfmlUtilities::getEvent ( )
```

give acceess to event

3.16.3.2 getWindow()

```
sf::RenderWindow * SfmlUtilities::getWindow ( )
```

give acceess to window

3.16.3.3 inputCheck()

```
void SfmlUtilities::inputCheck (
            bool & isGameStarted,
            bool & isGameEnded,
            bool & isNewGameStarted )
```

checks input

3.16.3.4 isWindowOpened()

```
const bool SfmlUtilities::isWindowOpened ( ) const
```

return information is program window open

The documentation for this class was generated from the following files:

- SfmlUtilities.h
- SfmlUtilities.cpp

3.17 Sounds Class Reference

```
#include <Sounds.h>
```

Public Member Functions

- Sounds ()
- ~Sounds ()
- sf::SoundBuffer * getSoundBuffer (string key)
- void stopMusic ()
- void playMusic ()
- void playGameOverSound ()
- void stopGameOverSound ()

3.17.1 Detailed Description

Declaration of sounds class

Author

Michal Pawlowski

Date

2021-05-11

3.17.2 Constructor & Destructor Documentation

3.17.2.1 Sounds()

```
Sounds::Sounds ( )
```

constructor

3.17.2.2 ~Sounds()

```
Sounds::~Sounds ( )
```

destructor

3.17.3 Member Function Documentation

3.17.3.1 getSoundBuffer()

```
sf::SoundBuffer * Sounds::getSoundBuffer (
            string key )
```

give access to texture pointer

3.17.3.2 playGameOverSound()

```
void Sounds::playGameOverSound ( )
```

play game over sound

3.17.3.3 playMusic()

```
void Sounds::playMusic ( )
```

start music

3.17.3.4 stopGameOverSound()

```
void Sounds::stopGameOverSound ( )
```

stop game over sound

3.17.3.5 stopMusic()

```
void Sounds::stopMusic ( )
```

stop music

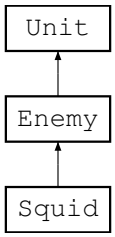
The documentation for this class was generated from the following files:

- Sounds.h
- Sounds.cpp

3.18 Squid Class Reference

```
#include <Squid.h>
```

Inheritance diagram for Squid:



Public Member Functions

- **Squid** (sf::Texture ∗crab1, sf::Texture ∗crab2, float x, float y)

Additional Inherited Members

3.18.1 Detailed Description

Declaration of Enemy class

Author

Michal Pawlowski

Date

2021-05-11

The documentation for this class was generated from the following files:

- Squid.h
- Squid.cpp

3.19 Textures Class Reference

`#include <Textures.h>`

Public Member Functions

- Textures ()
- ~Textures ()
- sf::Texture * getTexture (string key)
- sf::Font * getFont ()

3.19.1 Detailed Description

Declaration of Textures class

Author

Michal Pawlowski

Date

2021-05-11

3.19.2 Constructor & Destructor Documentation

3.19.2.1 Textures()

```
Textures::Textures ( )
```

no argument constructor

3.19.2.2 ~Textures()

```
Textures::~Textures ( )
```

desctructor

3.19.3 Member Function Documentation

3.19.3.1 getFont()

```
sf::Font * Textures::getFont ( )
```

give access to font pointer

3.19.3.2 getTexture()

```
sf::Texture * Textures::getTexture (
            string key )
```

give access to texture pointer

The documentation for this class was generated from the following files:

- Textures.h
- Textures.cpp

3.20 Timer Class Reference

```
#include <Timer.h>
```

Public Member Functions

- Timer ()
- ~Timer ()
- void CalculateDeltaTime ()
- float getDeltaTime ()

3.20.1 Detailed Description

Declaration of Timer class

Author

Michal Pawlowski

Date

2021-05-11

3.20.2 Constructor & Destructor Documentation

3.20.2.1 Timer()

```
Timer::Timer ( )
```

no argument constructor

3.20.2.2 ~Timer()

```
Timer::~Timer ( )
```

destructor

3.20.3 Member Function Documentation

3.20.3.1 CalculateDeltaTime()

```
void Timer::CalculateDeltaTime ( )
```

calculate time of frame duration

3.20.3.2 getDeltaTime()

```
float Timer::getDeltaTime ( )
```

returns deltaTime

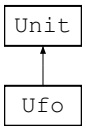
The documentation for this class was generated from the following files:

- Timer.h
- Timer.cpp

3.21 Ufo Class Reference

`#include <Ufo.h>`

Inheritance diagram for Ufo:



Public Member Functions

- Ufo (sf::Texture *ufoTexture, sf::SoundBuffer *ufoSoundParam)
- void update (float deltaTime)
- void reset ()
- void render (sf::RenderTarget *target)
- void stopUfoSound ()

Additional Inherited Members

3.21.1 Detailed Description

Declaration of Ufo class

Author

Michal Pawlowski

Date

2021-05-11

3.21.2 Constructor & Destructor Documentation

3.21.2.1 Ufo()

```
Ufo::Ufo (
            sf::Texture * ufoTexture,
            sf::SoundBuffer * ufoSoundParam )
```

constructor

3.21.3 Member Function Documentation

3.21.3.1 render()

```
void Ufo::render (
            sf::RenderTarget * target )
```

renders ufo current state

3.21.3.2 reset()

```
void Ufo::reset ( )
```

set ufo in default stage

3.21.3.3 stopUfoSound()

```
void Ufo::stopUfoSound ( )
```

stop ufo sound

3.21.3.4 update()

```
void Ufo::update (
            float deltaTime )
```

update ufo status

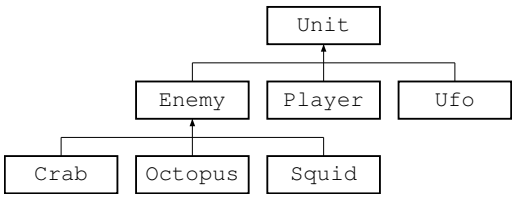
The documentation for this class was generated from the following files:

- Ufo.h
- Ufo.cpp

3.22 Unit Class Reference

```
#include <Unit.h>
```

Inheritance diagram for Unit:



Public Member Functions

- const sf::Vector2f & getPosition () const
- const sf::FloatRect getBounds () const
- virtual const bool isAttackPossible ()=0
- void render (sf::RenderTarget *target)

Protected Member Functions

- void initializeSprite (float xScale, float yScale)

Protected Attributes

- sf::Sprite sprite
- sf::Texture * texture
- float movementSpeed
- float timeToAttack

3.22.1 Detailed Description

Declaration of Unit virtual class

Author

Michal Pawlowski

Date

2021-05-11

3.22.2 Member Function Documentation

3.22.2.1 getBounds()

```
const sf::FloatRect Unit::getBounds ( ) const
```

give info about sprite coverage

3.22.2.2 getPosition()

```
const sf::Vector2f & Unit::getPosition ( ) const
```

give information about unit position

3.22.2.3 initializeSprite()

```
void Unit::initializeSprite (
            float xScale,
            float yScale )  [protected]
```

sprite initialization

3.22.2.4 isAttackPossible()

```
virtual const bool Unit::isAttackPossible ( )  [pure virtual]
```

inform is attack possible

Implemented in Enemy, and Player.

3.22.2.5 render()

```
void Unit::render (
            sf::RenderTarget * target )
```

renders object current state

3.22.3 Member Data Documentation

3.22.3.1 movementSpeed

```
float Unit::movementSpeed  [protected]
```

define movement speed

3.22.3.2 sprite

```
sf::Sprite Unit::sprite  [protected]
```

representation of a texture

3.22.3.3 texture

```
sf::Texture* Unit::texture  [protected]
```

texture object

3.22.3.4 timeToAttack

```
float Unit::timeToAttack  [protected]
```

time to next attack

The documentation for this class was generated from the following files:

- Unit.h
- Unit.cpp

3.23 Wall Class Reference

```
#include <Wall.h>
```

Public Member Functions

- Wall (float xPos, sf::Texture ∗square, sf::Texture ∗triangle)
- vector< Block > & getWall ()
- void renderWall (sf::RenderWindow ∗window) const

3.23.1 Detailed Description

Declaration of Wall class

Author

Michal Pawlowski

Date

2021-05-11

3.23.2 Constructor & Destructor Documentation

3.23.2.1 Wall()

```
Wall::Wall (
            float xPos,
            sf::Texture * square,
            sf::Texture * triangle )
```

constructor

3.23.3 Member Function Documentation

3.23.3.1 getWall()

```
vector< Block > & Wall::getWall ( )
```

give acces to vector of Blocks

3.23.3.2 renderWall()

```
void Wall::renderWall (
            sf::RenderWindow * window ) const
```

render single wall

The documentation for this class was generated from the following files:

- Wall.h
- Wall.cpp

3.24 Walls Class Reference

```
#include <Walls.h>
```

Public Member Functions

- Walls (sf::Texture *square1, sf::Texture *square2, sf::Texture *square3, sf::Texture *square4, sf::Texture *triangle1, sf::Texture *triangle2, sf::Texture *triangle3, sf::Texture *triangle4)
- void rebuiltWalls ()
- vector< Wall > & getWalls ()
- map< string, sf::Texture * > & getTextures ()
- void renderWalls (sf::RenderWindow *window)

3.24.1 Detailed Description

Declaration of Walls class

Author

Michal Pawlowski

Date

2021-05-11

3.24.2 Constructor & Destructor Documentation

3.24.2.1 Walls()

```
Walls::Walls (
            sf::Texture * square1,
            sf::Texture * square2,
            sf::Texture * square3,
            sf::Texture * square4,
            sf::Texture * triangle1,
            sf::Texture * triangle2,
            sf::Texture * triangle3,
            sf::Texture * triangle4 )
```

construcor

3.24.3 Member Function Documentation

3.24.3.1 getTextures()

```
map< string, sf::Texture * > & Walls::getTextures ( )
```

give acces to vector with textures

3.24.3.2 getWalls()

```
vector< Wall > & Walls::getWalls ( )
```

give acces to vector with walls

3.24.3.3 rebuiltWalls()

```
void Walls::rebuiltWalls ( )
```

delete and the create new walls

3.24.3.4 renderWalls()

```
void Walls::renderWalls (
            sf::RenderWindow * window )
```

render all walls

The documentation for this class was generated from the following files:

- Walls.h
- Walls.cpp