Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

# Programowanie Komputerów

## Sprawozdanie z projektu

| | |
|---|---|
| autorzy | Tomasz Zawadzki<br>Wojciech Siudy |
| prowadzący | dr inż. Jolanta Kawulok |
| rok akademicki | 2021/2022 |
| kierunek | Informatyka |
| rodzaj studiów | SSI |
| semestr | 4 |
| sekcja | 41 |
| termin oddania sprawozdania | 2022-06-24 |

Program jest grą inspirowaną klasycznym tytułem “Super Mario Bros”. Interfejs graficzny został oparty na bibliotece SFML. Program jest wieloplatformowy i zbudowany w oparciu o paradygmat obiektowy. Zastosowane w nim są wzorce projektowe oraz metody polimorficzne.

1. **Opis działania programu z perspektywy użytkownika.**

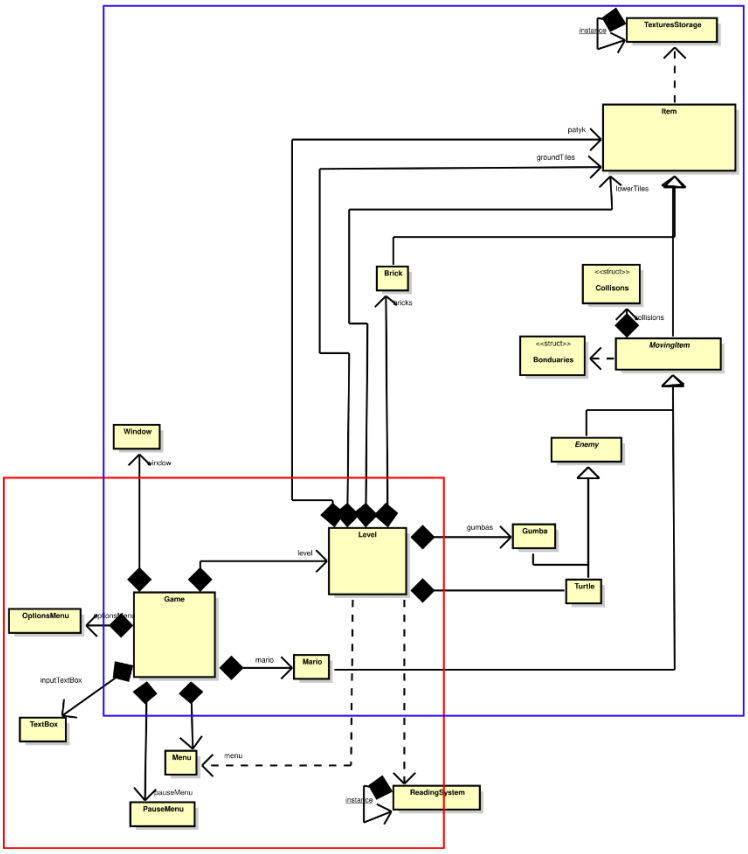
Program uruchomi się w oknie aplikacji. Do nawigacji wykorzystana jest klawiatura. Samo kontrolowanie postaci podczas gry również odbywa się poprzez naciśnięcie odpowiednich przycisków. Grę da się zatrzymać, zapisać lub rozpocząć od nowa. Możliwe jest wczytywanie różnych poziomów poprzez podanie nazwy pliku przechowującego o nich. Najwyższy osiągnięty wynik jest zapisywany na dysku komputera.

2. **Zasady gry.**

Jest to gra platformowa polegająca na ominięciu różnych fal przeciwników oraz przeskakiwaniu przepaści, w celu dotarcia do flagi na końcu poziomu. Gracz posiada trzy życia, po wyczerpaniu których przegrywa i może rozpocząć grę od nowa. Punkty zyskiwane są poprzez zabijanie wrogów lub uderzenie głową w cegłę przechowującą monetę.

## 3. Struktura klas i schemat działania programu

Kolorem czerwonym oznaczono obszar działalności Tomasza Zawadzkiego, a niebieskim Wojciecha Siudego.




Rys 2. Diagram hierarchii klas

W głównej funkcji programu utworzony będzie obiekt klasy `Game` - silnik gry, a następnie `Window` - menedżer otwartego okna. Rozpoczęcie nowej gry utworzy obiekt `Mario` oraz `Level`, który będzie mieścił mapę, zawierającą obiekty pozostałych klas oraz obsługujący relacje między nimi.

## 4. Funkcje dostępne dla użytkownika

Po uruchomieniu programu użytkownikowi zostanie wyświetlone cieszące oko menu. Do wyboru będzie parę funkcji:

- Nowa gra

- Kontynuuj
- Ustawienia
- Wyjdź z gry

Po wybraniu opcji Nowa gra, użytkownik rozpocznie rozgrywkę od zera.
Wybierając opcję Kontynuuj zostanie wczytany z pliku ostatni stan rozgrywki, a gra rozpocznie się tam, gdzie poprzednio się zakończyła.
W ustawieniach będzie można zmienić podstawowe parametry gry, dostosowując je do własnych preferencji.
W zakładce twórcy znajdą się informacje o autorach projektu.
Po wyborze Wyjdź z gry okno z grą zostanie zamknięte, a stan gry zapisany.

## 5. Użyte biblioteki omawiane na zajęciach

a. <thread> (wielowątkowość prosta – std::thread) - cztery możliwe kolizje każdego z poruszających się elementów są analizowane równolegle [WS]
b. <thread> (wielowątkowość złożona - std::jthread, std::lock_guard) - obiekt przechowujący tekstury wczytane z pliku jest zaimplementowany jako singleton z uwzględnieniem wyścigu dostępu do zasobów. Niektóre wątki w klasie Level są *joinowane* automatycznie [WS]
c. <filesystem> + <ranges> - mechanizm poszukiwania pliku z zapisanym najwyższym wynikiem korzysta z rekursywnego iteratora plików umieszczonego w pętli for_each wywołującej wyrażenie lambda dla każdego elementu [WS]
d. <regex> - biblioteka regex użyta jest w celu automatycznego znajdowania plików z poziomami. Jeśli plik zaczyna się frazą „Level” lub „level”, a kończy rozszerzeniem „.txt”, wczytywany jest automatycznie jako jeden z poziomów domyślnych wczytywanych przy uruchomieniu gry. Użycie biblioteki regex ma także zastosowanie w sprawdzaniu poprawności formatu pliku wczytanego przez użytkownika. Jeśli plik kończy się jakimkolwiek innym rozszerzeniem niż „.txt”, zostanie odrzucony, a wczytany zostanie jeden z poziomów domyślnych. [TZ]
e. <filesystem> - Użycie biblioteki filesystem ma swoje zastosowanie w połączeniu z biblioteką regex. Przeszukuje ona wszystkie foldery gry w poszukiwaniu plików, po czym zapisuje ich nazwy do odpowiedniego wektora. [TZ]

f. <ranges> - biblioteka ranges użyta jest w celu iteracji przez wszystkie elementy wektora z nazwami poziomów, oraz filtrację tych poprawnych, co osiągnięte jest w połączeniu z biblioteką <regex>. [TZ]
g. <thread> - (wielowątkowość złożona – std::jthread, std::lock_guard) biblioteka użyta jest podczas wczytywania poziomów w konstruktorze klasy Level. Pozwala to na wykonanie wielu niezależnych operacji jednocześnie, co przyspiesza proces tworzenia poziomu. Użyta jest także w klasie ReadingSystem, która jest Singletonem, co pozwala zapobiec wyścigowi po zasoby. [TZ]

6. **Poprawność działania.**

Program kompiluje się zarówno w trybie debug, jak i release - niezależnie od platformy uruchomieniowej. Upewniono się o braku wycieków pamięci przy użyciu narzędzia Valgrind.

7. **Szczegółowy opis metod i klas znajduje się w załączniku.**
8. **Wnioski**

Tworzenie gry za pomocą biblioteki SFML pozwoliło nam na poznanie koncepcji tworzenia gier komputerowych. Zaimplementowanie bibliotek prezentowanych na zajęciach umożliwiło lepsze zrozumienie nowych możliwości języka c++ oraz ich sprawne wykorzystywanie. Największym wyzwaniem w projekcie okazało się dołączenie zewnętrznych bibliotek, co z początku powodowało wiele problemów. Poznane na Inżynierii oprogramowania wzorce projektowe pozwoliły także usprawnic projekt poprzez ich implementacje.

# Super Mario

Wygenerowano przez Doxygen 1.8.17

# Rozdział 1

# Indeks hierarchiczny

## 1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

# Rozdział 2

# Indeks klas

## 2.1 Lista klas

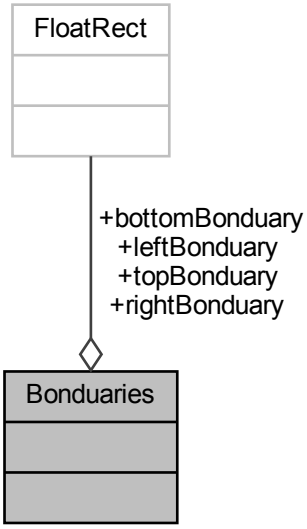
Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

# Rozdział 3

# Dokumentacja klas

## 3.1 Dokumentacja struktury Bonduaries

Diagram współpracy dla Bonduaries:




### Atrybuty publiczne

- sf::FloatRect **leftBonduary**
- sf::FloatRect **rightBonduary**
- sf::FloatRect **topBonduary**
- sf::FloatRect **bottomBonduary**

Dokumentacja dla tej struktury została wygenerowana z pliku:

- /home/wojciech/CLionProjects/super_mario/src/headers/Item.h

## 3.2 Dokumentacja klasy Brick

Diagram dziedziczenia dla Brick

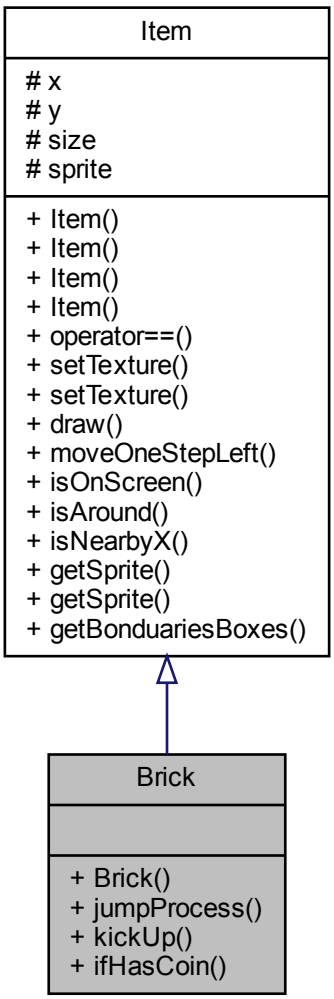
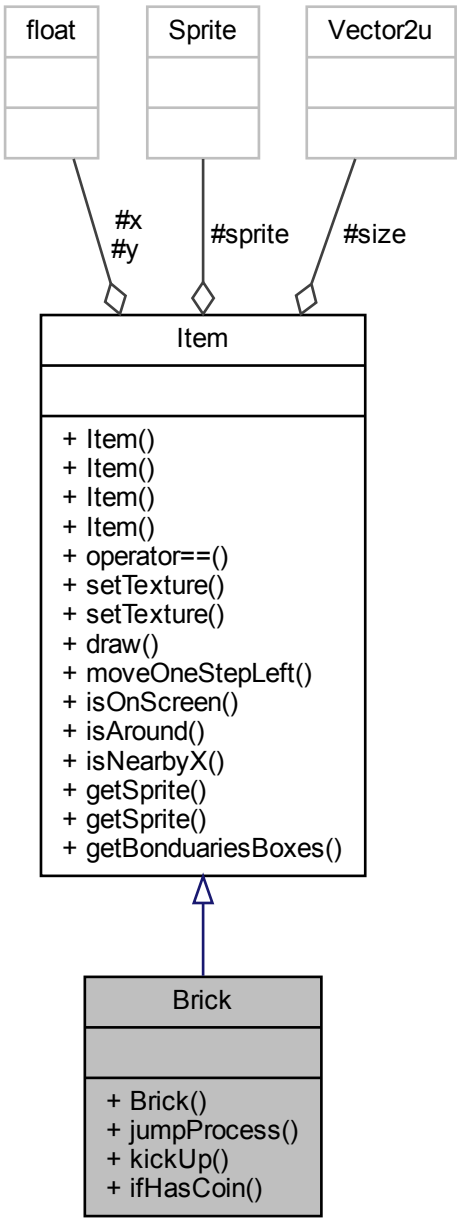


Diagram współpracy dla Brick:




## Metody publiczne

- **Brick** (float x, float y)
- void jumpProcess ()
- void kickUp ()
- bool ifHasCoin ()

### Dodatkowe Dziedziczone Składowe

### 3.2.1 Dokumentacja funkcji składowych

#### 3.2.1.1 ifHasCoin()

```
bool Brick::ifHasCoin ( )  [inline]
```

Zwraca informacje o przechowywaniu monety.

#### 3.2.1.2 jumpProcess()

```
void Brick::jumpProcess ( )
```

Przetwarza ewentualne podrzucenie.

#### 3.2.1.3 kickUp()

```
void Brick::kickUp ( )
```

Wyzwala podrzucenie po uderzeniu weń głową gracza.

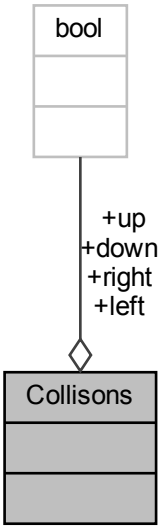
Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Brick.h
- /home/wojciech/CLionProjects/super_mario/src/Brick.cpp

## 3.3 Dokumentacja struktury Collisons

```
#include <MovingItem.h>
```

Diagram współpracy dla Collisons:



### Atrybuty publiczne

- bool **left** = false
- bool **right** = false
- bool **up** = false
- bool **down** = false

### 3.3.1 Opis szczegółowy

Struktura przechowująca informację o możliwościach ruchu

Dokumentacja dla tej struktury została wygenerowana z pliku:

- /home/wojciech/CLionProjects/super_mario/src/headers/MovingItem.h

## 3.4 Dokumentacja klasy Enemy

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

Diagram dziedziczenia dla Enemy

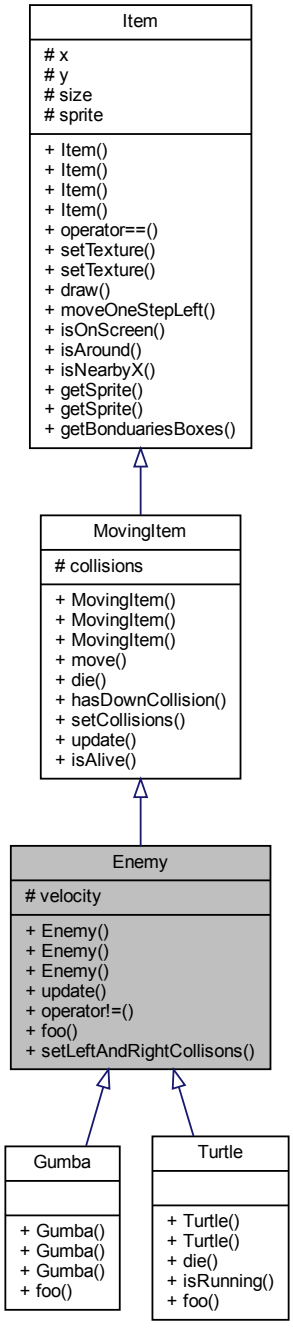
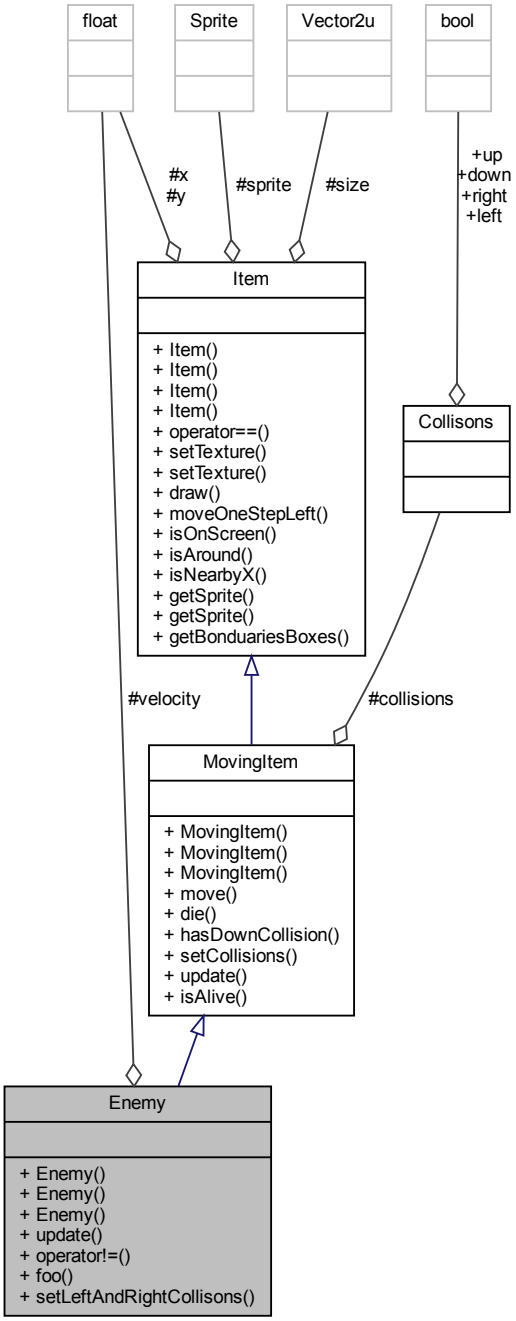


Diagram współpracy dla Enemy:




## Metody publiczne

- **Enemy** (const Enemy &other)=default
- **Enemy** (int iX, int iY)
- void update ()
- bool operator!= (const Enemy &rEnemy)
- virtual void foo ()=0
- void setLeftAndRightCollisons (bool left, bool right)

### Atrybuty chronione

- float **velocity**

## 3.4.1 Opis szczegółowy

Klasa reprezentuje wroga

## 3.4.2 Dokumentacja funkcji składowych

#### 3.4.2.1 foo()

```
virtual void Enemy::foo ( )  [pure virtual]
```

Pusta metoda definiująca niniejszą klasę jako abstrakcyjną.

Implementowany w Gumba i Turtle.

#### 3.4.2.2 operator"!=()

```
bool Enemy::operator!= (
            const Enemy & rEnemy )
```

Operator porównania. Bierze pod uwagę tylko pozycję.

#### 3.4.2.3 setLeftAndRightCollisons()

```
void Enemy::setLeftAndRightCollisons (
            bool left,
            bool right )
```

Ustawia lewą i prawą kolizję

**Parametry**

| | |
|---|---|
| *left* | lewa kolizja |
| *right* | prawa kolizja |

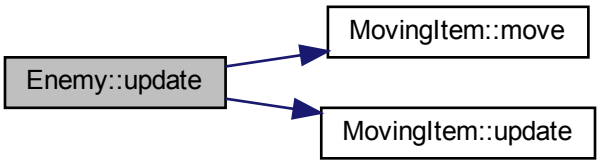
#### 3.4.2.4 update()

```
void Enemy::update ( )  [virtual]
```

Zwraca informację, czy dany wróg powinien umrzeć. Należy wykorzystać przy iteracji w kolejnej klatce. Rozszerza metodę klasy MovingItem o automatyczny ruch i odbicia

Reimplementowana z MovingItem.

Oto graf wywołań dla tej funkcji:



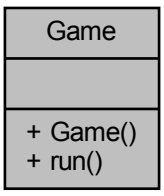

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Enemy.h
- /home/wojciech/CLionProjects/super_mario/src/Enemy.cpp

## 3.5 Dokumentacja klasy Game

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

Diagram współpracy dla Game:




### Metody publiczne

- void run ()

### 3.5.1 Opis szczegółowy

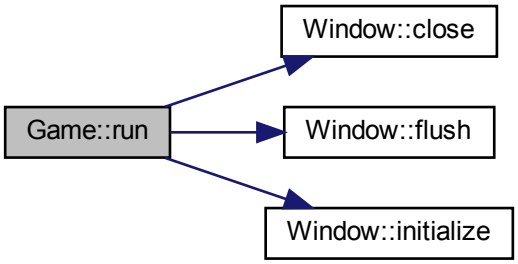
Silnik gry

### 3.5.2 Dokumentacja funkcji składowych

#### 3.5.2.1 run()

```
void Game::run ( )
```

Głowna pętla programu Oto graf wywołań dla tej funkcji:




Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Game.h
- /home/wojciech/CLionProjects/super_mario/src/Game.cpp

## 3.6 Dokumentacja klasy Gumba

```
#include <Gumba.h>
```

Diagram dziedziczenia dla Gumba

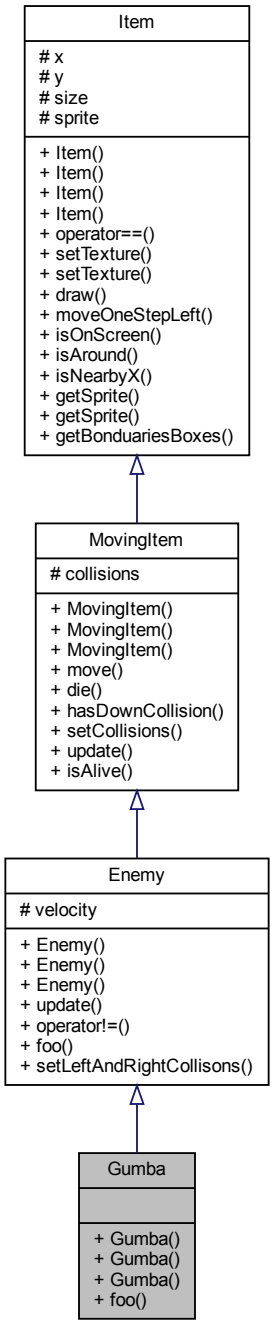
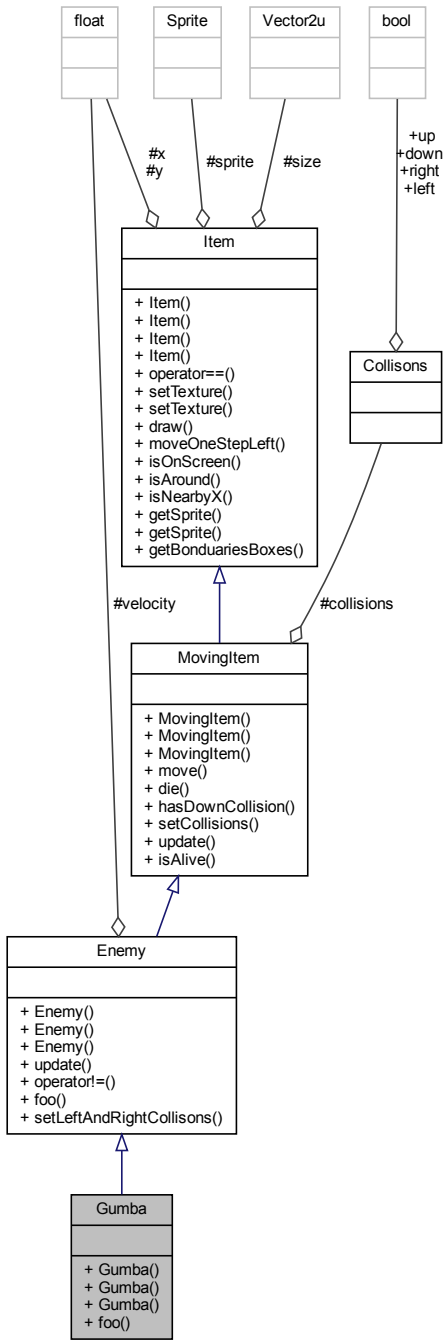


Diagram współpracy dla Gumba:




## Metody publiczne

- **Gumba** (const Gumba &)=default
- **Gumba** (int iX, int iY)
- void foo () override

### Dodatkowe Dziedziczone Składowe

### 3.6.1 Opis szczegółowy

Klasa reprezentuje przeciwnika "Gumba"

### 3.6.2 Dokumentacja funkcji składowych

#### 3.6.2.1 foo()

```
void Gumba::foo ( )  [inline], [override], [virtual]
```

Implementacja interfejsu Enemy "pro forma"

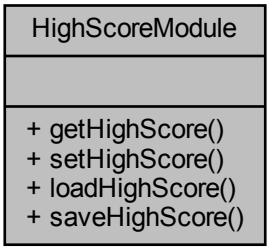
Implementuje Enemy.

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Gumba.h
- /home/wojciech/CLionProjects/super_mario/src/Gumba.cpp

## 3.7 Dokumentacja klasy HighScoreModule

Diagram współpracy dla HighScoreModule:



### Statyczne metody publiczne

- static int **getHighScore** ()
- static void **setHighScore** (int newScore)
- static void **loadHighScore** ()
- static void **saveHighScore** ()

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/HighScoreModule.h
- /home/wojciech/CLionProjects/super_mario/src/HighScoreModule.cpp

## 3.8 Dokumentacja klasy Item

```
#include <Item.h>
```

Diagram dziedziczenia dla Item

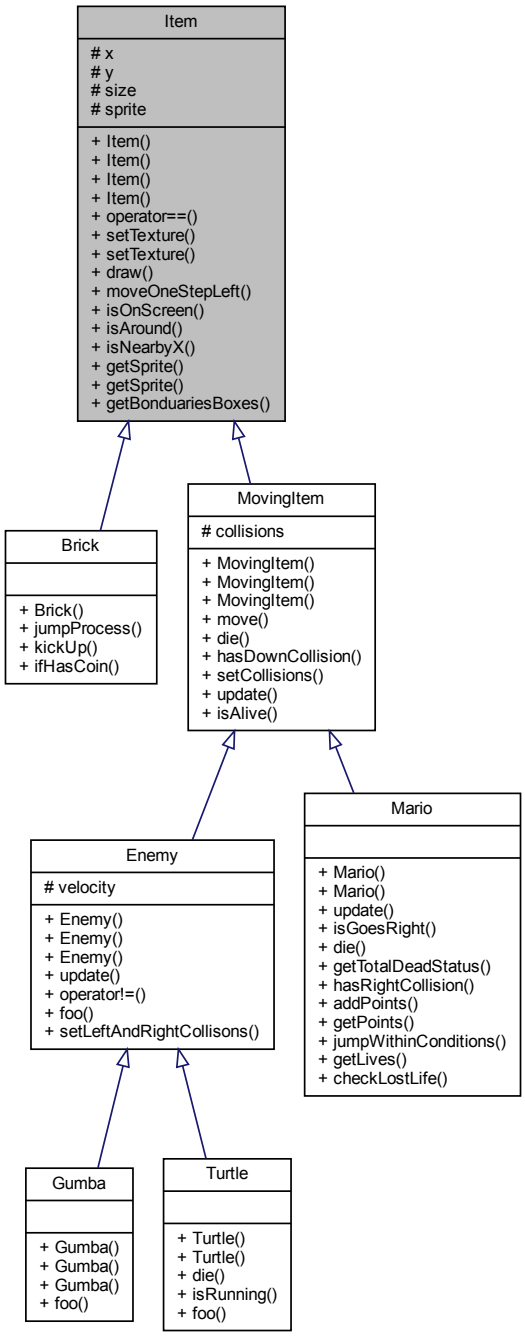
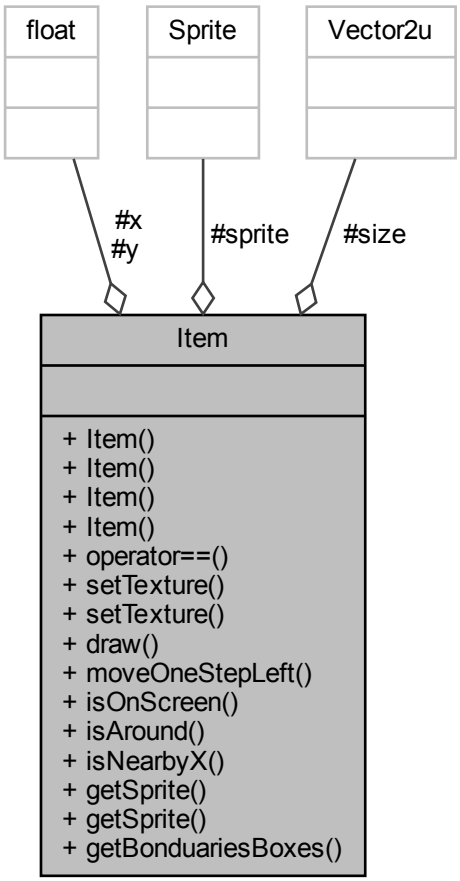


Diagram współpracy dla Item:




## Metody publiczne

- Item (int x, int y)
- **Item** (int x, int y, const std::shared_ptr< sf::Texture > &t)
- **Item** (const Item &other)
- bool operator== (const Item &rItem)
- void setTexture (const std::string &s)
- void setTexture (const std::shared_ptr< sf::Texture > &t)
- void draw (sf::RenderWindow &iWindow)
- void moveOneStepLeft ()
- bool isOnScreen () const
- bool isAround (float ix) const
- bool isNearbyX (const Item &other) const
- const sf::Sprite & getSprite () const
- sf::Sprite getSprite ()
- Bonduaries getBonduariesBoxes ()

## Atrybuty chronione

- float **x** {}
- float **y** {}
- sf::Vector2u **size**
- sf::Sprite **sprite**

### 3.8.1 Opis szczegółowy

Klasa przechowuje informacje o obiekcie w grze

### 3.8.2 Dokumentacja konstruktora i destruktora

#### 3.8.2.1 Item()

```
Item::Item (
            int x,
            int y )
```

Umieszcza na ekranie element w zadanym miejscu.

**Parametry**

| | |
|---|---|
| *x* | |
| *y* | |

Oto graf wywołań dla tej funkcji:


### 3.8.3 Dokumentacja funkcji składowych

#### 3.8.3.1 draw()

```
void Item::draw (
            sf::RenderWindow & iWindow )
```

Wyświetla element we wskazanym oknie.

**Parametry**

| | |
|---|---|
| *iWindow* | okno |

#### 3.8.3.2 getBonduariesBoxes()

```
Bonduaries Item::getBonduariesBoxes ( )
```

Zwraca prostokąty obwiedni do kontroli.

**Zwraca**

#### 3.8.3.3 getSprite() [1/2]

```
sf::Sprite Item::getSprite ( )  [inline]
```

Zwraca sprite.

#### 3.8.3.4 getSprite() [2/2]

```
const sf::Sprite & Item::getSprite ( ) const
```

Zwraca sprite.

#### 3.8.3.5 isAround()

```
bool Item::isAround (
            float ix ) const  [inline]
```

Metoda zwracająca informację, czy element znajduje się w otoczeniu punktu.

**Parametry**

| | |
|---|---|
| *ix* | położenie na osi poziomej |

#### 3.8.3.6 isNearbyX()

```
bool Item::isNearbyX (
            const Item & other ) const  [inline]
```

Metoda zwracająca informację, czy wskazany element znajduje się w pobliżu this->. Porównywana jest wartość położenia na osi poziomej.

**Parametry**

| | |
|---|---|
| *other* | inny element |

#### 3.8.3.7 isOnScreen()

```
bool Item::isOnScreen ( ) const  [inline]
```

Metoda filtrująca iteracje

**Zwraca**

czy analiza danego elementu ma sens

#### 3.8.3.8 moveOneStepLeft()

```
void Item::moveOneStepLeft ( )
```

Przemieszcza wszystkie elementy o jednen krok w lewo

#### 3.8.3.9 operator==()

```
bool Item::operator== (
            const Item & rItem )
```

Zwraca informacje o zetknięciu się elementów Oto graf wywołań dla tej funkcji:

#### 3.8.3.10 setTexture() [1/2]

```
void Item::setTexture (
            const std::shared_ptr< sf::Texture > & t )
```

Po zadaniu tekstury ustawia też wymiary i środek.

**Parametry**

| | |
|---|---|
| *t* | wskaźnik na teksturę |

#### 3.8.3.11 setTexture() [2/2]

```
void Item::setTexture (
            const std::string & s )
```

Po zadaniu tekstury ustawia też wymiary i środek.

**Parametry**

| | |
|---|---|
| *s* | ścieżka do tekstury |

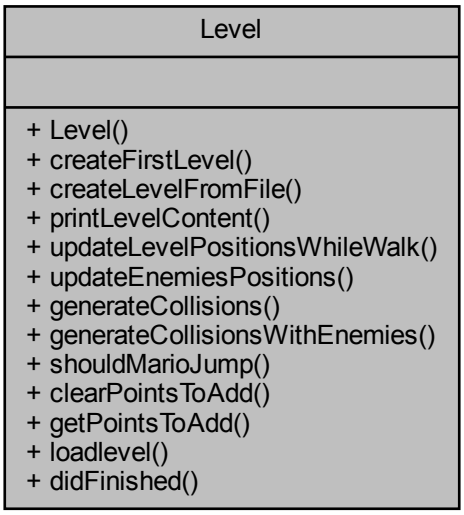
Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Item.h
- /home/wojciech/CLionProjects/super_mario/src/Item.cpp

## 3.9 Dokumentacja klasy Level

```
#include <Level.h>
```

Diagram współpracy dla Level:




## Metody publiczne

- void **createFirstLevel** ()
- void **createLevelFromFile** (std::string &path)
- void printLevelContent (sf::RenderWindow &iwindow)
- void updateLevelPositionsWhileWalk ()
- void updateEnemiesPositions ()
- void generateCollisions (MovingItem &movingItem, bool headFlag=false)
- void generateCollisionsWithEnemies (MovingItem &mario)
- bool shouldMarioJump ()
- void clearPointsToAdd ()
- int getPointsToAdd ()
- void **loadlevel** ()
- bool didFinished (const MovingItem &mario)

### 3.9.1 Opis szczegółowy

Klasa zawiera aktualny poziom gry

### 3.9.2 Dokumentacja funkcji składowych

#### 3.9.2.1 clearPointsToAdd()

```
void Level::clearPointsToAdd ( )  [inline]
```

Metoda czyszcząca bufor punktów z analizy w obecnej klatce.

#### 3.9.2.2 didFinished()

```
bool Level::didFinished (
            const MovingItem & mario )
```

Zwraca informację czy poziom został ukończony

**Parametry**

| | |
|---|---|
| *mario* | gracz |

#### 3.9.2.3 generateCollisions()

```
void Level::generateCollisions (
            MovingItem & movingItem,
            bool headFlag = false )
```

Ustawia parametry kolizji dla danego obiektu względem pozostałych elementów

**Parametry**

| | |
|---|---|
| *movingItem* | modyfikowany obiekt |
| *headFlag* | czy powinien wykonać akcję głową |

Oto graf wywołań dla tej funkcji:


#### 3.9.2.4 generateCollisionsWithEnemies()

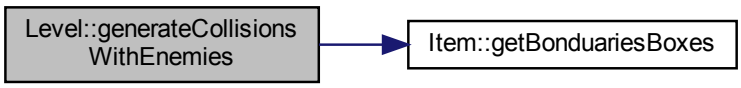
```
void Level::generateCollisionsWithEnemies (
            MovingItem & mario )
```

Wyzwala akcję zwoiązane z kolizjami Maria z wrogami. Zabija wrogów lub Mario.

**Parametry**

| | |
|---|---|
| *mario* | Mario |

Oto graf wywołań dla tej funkcji:




#### 3.9.2.5 getPointsToAdd()

```
int Level::getPointsToAdd ( )  [inline]
```

Metoda zwracająca punkty zdobyte w danej klatce.

#### 3.9.2.6 printLevelContent()

```
void Level::printLevelContent (
            sf::RenderWindow & iwindow )
```

Renderuje elementy poziomu na ekranie

**Parametry**

| | |
|---|---|
| *iwindow* | okno |

#### 3.9.2.7 shouldMarioJump()

```
bool Level::shouldMarioJump ( )  [inline]
```

Metoda sprawdzająca, czy Mario powinien być podrzucony do góry.

#### 3.9.2.8 updateEnemiesPositions()

```
void Level::updateEnemiesPositions ( )
```

Kontynuuje ruch wrogich postaci Oto graf wywołań dla tej funkcji:


#### 3.9.2.9 updateLevelPositionsWhileWalk()

```
void Level::updateLevelPositionsWhileWalk ( )
```

Przemieszcza cały świat w lewo (oprócz Maria) ruch odbywa się względem niego

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Level.h
- /home/wojciech/CLionProjects/super_mario/src/Level.cpp

## 3.10 Dokumentacja klasy Mario

```
#include <Mario.h>
```

Diagram dziedziczenia dla Mario

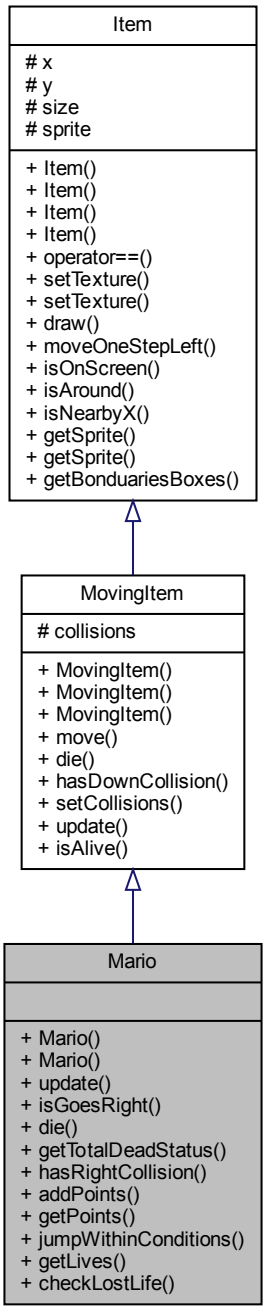
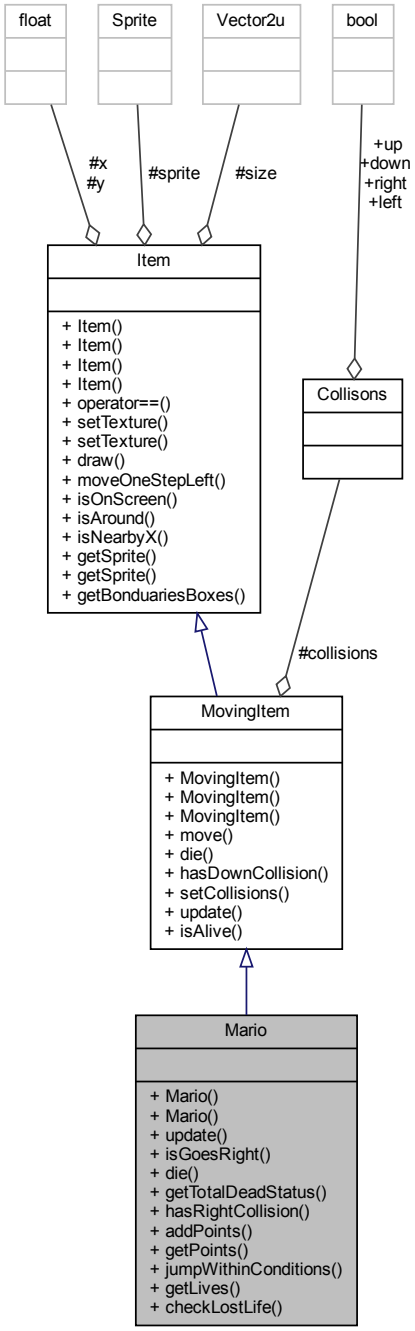


Diagram współpracy dla Mario:




## Metody publiczne

- **Mario** (int prevLives)
- void update () override
- bool isGoesRight () const
- void die () override
- bool **getTotalDeadStatus** ()

- bool hasRightCollision ()
- void addPoints (int points)
- int **getPoints** ()
- void **jumpWithinConditions** ()
- int **getLives** () const
- bool **checkLostLife** ()

## Dodatkowe Dziedziczone Składowe

## 3.10.1 Opis szczegółowy

Klasa przechowuje dane o głównej postaci gry

## 3.10.2 Dokumentacja funkcji składowych

### 3.10.2.1 addPoints()

```
void Mario::addPoints (
            int points )
```

Dodaje punkty dla gracza

**Parametry**

| | |
|---|---|
| *points* | ilość punktów |

### 3.10.2.2 die()

```
void Mario::die ( )  [override], [virtual]
```

Rozpoczyna proces umierania gracza

Reimplementowana z MovingItem.

### 3.10.2.3 hasRightCollision()

```
bool Mario::hasRightCollision ( )  [inline]
```

Wykrywanie kolizji po prawej stronie

**Zwraca**

#### 3.10.2.4 isGoesRight()

```
bool Mario::isGoesRight ( ) const
```

Zwraca informację o ewentualnym ruchu w prawo, za połowę ekranu.

**Zwraca**

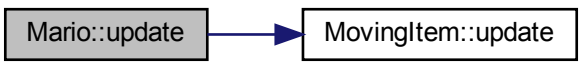
#### 3.10.2.5 update()

```
void Mario::update ( )  [override], [virtual]
```

Aktualizuje stan - metoda najwyższego poziomu abstrakcji dla klasy.

Reimplementowana z MovingItem.

Oto graf wywołań dla tej funkcji:



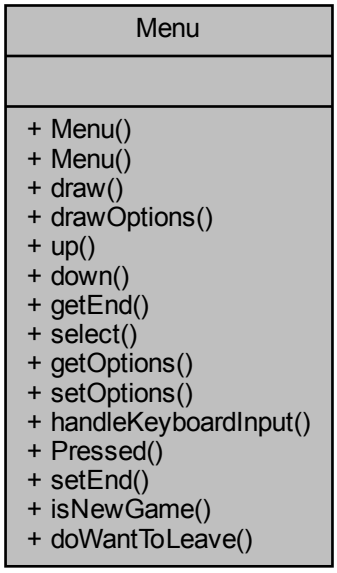

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Mario.h
- /home/wojciech/CLionProjects/super_mario/src/Mario.cpp

## 3.11 Dokumentacja klasy Menu

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

Diagram współpracy dla Menu:




## Metody publiczne

- **Menu** (float width, float height)
- void draw (sf::RenderWindow &window)
- void drawOptions (sf::RenderWindow &window)
- void **up** ()
- void **down** ()
- bool **getEnd** ()
- void **select** ()
- bool **getOptions** ()
- void **setOptions** ()
- void **handleKeyboardInput** (sf::RenderWindow &window)
- int **Pressed** ()
- void **setEnd** ()
- bool **isNewGame** () const
- bool **doWantToLeave** () const

### 3.11.1 Opis szczegółowy

Klasa zapewniająca opcje wyboru

### 3.11.2 Dokumentacja funkcji składowych

#### 3.11.2.1 draw()

```
void Menu::draw (
             sf::RenderWindow & window )
```

Metoda wyświetla menu na ekranie.

#### 3.11.2.2 drawOptions()

```
void Menu::drawOptions (
             sf::RenderWindow & window )
```

Do góry...

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Menu.h
- /home/wojciech/CLionProjects/super_mario/src/Menu.cpp

## 3.12 Dokumentacja klasy MovingItem

```
#include <MovingItem.h>
```

Diagram dziedziczenia dla MovingItem

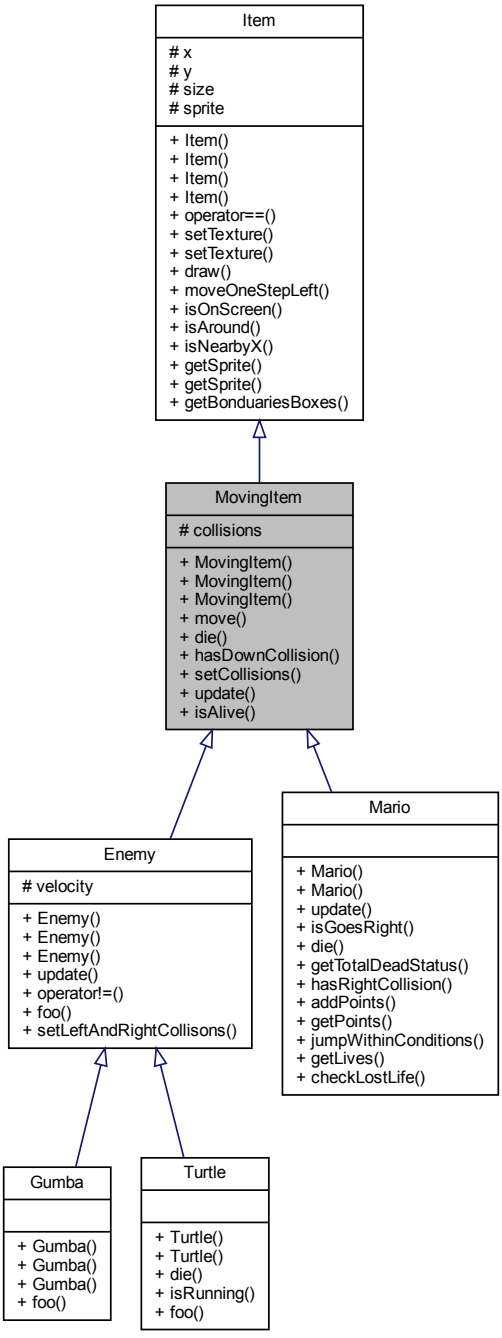
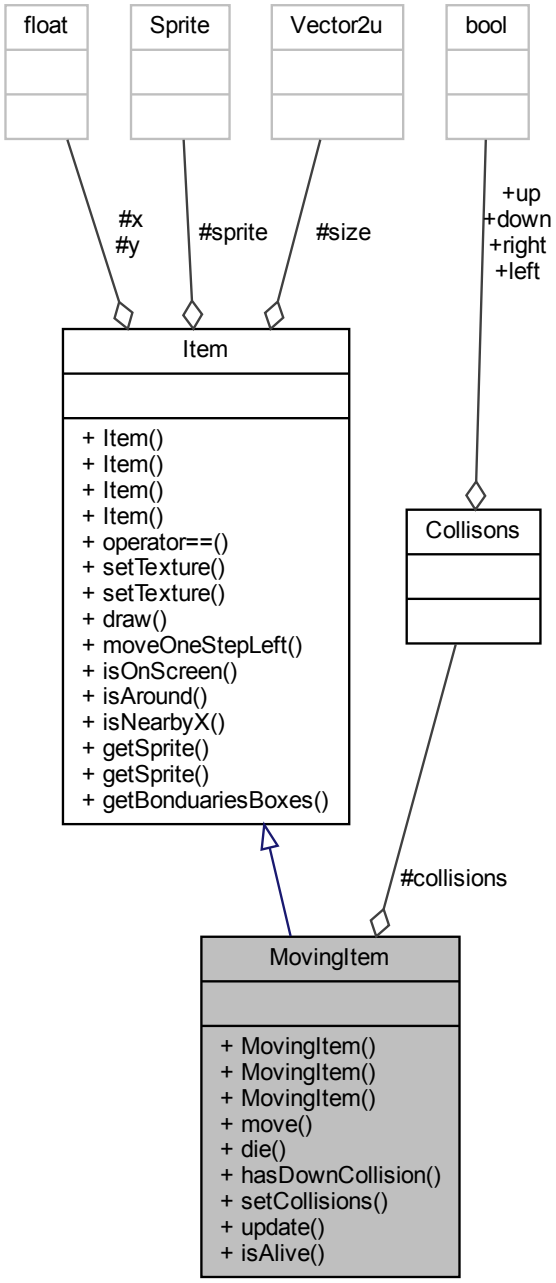


Diagram współpracy dla MovingItem:




## Metody publiczne

- MovingItem (const MovingItem &other)=default
- MovingItem (int iX, int iY)
- void move (float x, float y)
- virtual void **die** ()
- bool hasDownCollision () const

- void **setCollisions** (Collisons newCollisons)
- virtual void update ()
- bool **isAlive** () const

## Atrybuty chronione

- Collisons **collisions**

### 3.12.1 Opis szczegółowy

Klasa przechowuje informacje o poruszającym się obiekcie

### 3.12.2 Dokumentacja konstruktora i destruktora

#### 3.12.2.1 MovingItem() [1/2]

```
MovingItem::MovingItem (
            const MovingItem & other )  [default]
```

Konstruktor przenoszący

#### 3.12.2.2 MovingItem() [2/2]

```
MovingItem::MovingItem (
            int iX,
            int iY )
```

Konstruktor dwuargumentowy

### 3.12.3 Dokumentacja funkcji składowych

#### 3.12.3.1 hasDownCollision()

```
bool MovingItem::hasDownCollision ( ) const
```

Metoda zwracająca informację o dolnej kolizji.

**Zwraca**

czy występuje kolizja dolna

#### 3.12.3.2 move()

```
void MovingItem::move (
            float x,
            float y )
```

Przesuwa obiekt, o ile jest taka możliwość

**Parametry**

| | |
|---|---|
| *x* | przesunięcie poziome |
| *y* | przesunięcie pionowe |

#### 3.12.3.3 update()

```
void MovingItem::update ( )  [virtual]
```

Aktualizuje położenie i stan

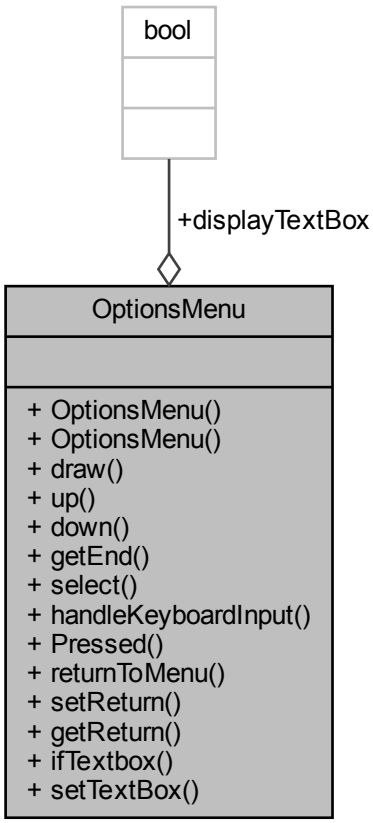
Reimplementowana w Mario i Enemy.

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/MovingItem.h
- /home/wojciech/CLionProjects/super_mario/src/MovingItem.cpp

## 3.13 Dokumentacja klasy OptionsMenu

Diagram współpracy dla OptionsMenu:



### Metody publiczne

- **OptionsMenu** (float width, float height)
- void **draw** (sf::RenderWindow &window)
- void **up** ()
- void **down** ()
- bool **getEnd** ()
- void **select** ()
- void **handleKeyboardInput** (sf::RenderWindow &window)
- int **Pressed** ()
- void **returnToMenu** ()
- void **setReturn** ()
- bool **getReturn** ()
- bool **ifTextbox** ()
- void **setTextBox** ()

### Atrybuty publiczne

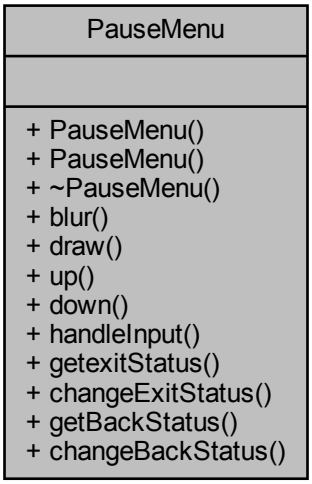
- bool **displayTextBox** = false

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/OptionsMenu.h
- /home/wojciech/CLionProjects/super_mario/src/OptionsMenu.cpp

## 3.14 Dokumentacja klasy PauseMenu

Diagram współpracy dla PauseMenu:



### Metody publiczne

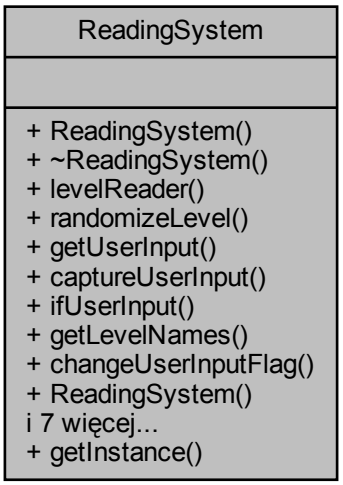
- **PauseMenu** (float width, float height)
- void **blur** (sf::RenderWindow &window)
- void **draw** (sf::RenderWindow &window)
- void **up** ()
- void **down** ()
- void **handleInput** ()
- bool **getexitStatus** ()
- void **changeExitStatus** ()
- bool **getBackStatus** ()
- void **changeBackStatus** ()

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/PauseMenu.h
- /home/wojciech/CLionProjects/super_mario/src/PauseMenu.cpp

## 3.15 Dokumentacja klasy ReadingSystem

Diagram współpracy dla ReadingSystem:




### Metody publiczne

- void **levelReader** ()
- void **randomizeLevel** ()
- std::string **getUserInput** ()
- void **captureUserInput** (std::string ui)

- bool **ifUserInput** ()
- std::vector< std::string > **getLevelNames** ()
- void **changeUserInputFlag** ()
- **ReadingSystem** (ReadingSystem &other)=delete
- void **operator=** (const ReadingSystem &)=delete
- void **findFilePath** ()
- bool **ifFirstRun** ()
- void **changeFirstRun** ()
- std::string **getFilePath** ()
- bool **ifFirstInput** ()
- void **changeFirstInput** ()

### Statyczne metody publiczne

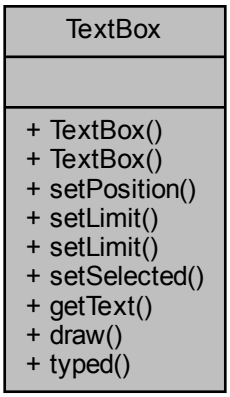
- static std::shared_ptr< ReadingSystem > **getInstance** ()

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/ReadingSystem.h
- /home/wojciech/CLionProjects/super_mario/src/ReadingSystem.cpp

## 3.16 Dokumentacja klasy TextBox

Diagram współpracy dla TextBox:



### Metody publiczne

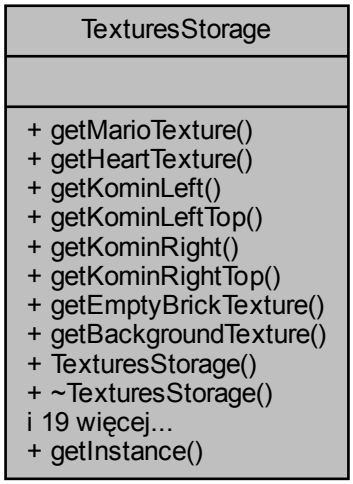
- **TextBox** (int size, sf::Color color, bool sel)
- void **setPosition** (sf::Vector2f pos)
- void **setLimit** (bool ToF)
- void **setLimit** (bool ToF, int lim)
- void **setSelected** (bool sel)
- std::string **getText** ()
- void **draw** (sf::RenderWindow &window)
- void **typed** (sf::Event ev)

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/TextBox.h
- /home/wojciech/CLionProjects/super_mario/src/TextBox.cpp

## 3.17 Dokumentacja klasy TexturesStorage

Diagram współpracy dla TexturesStorage:




### Metody publiczne

- const std::shared_ptr< sf::Texture > & **getMarioTexture** () const
- const std::shared_ptr< sf::Texture > & **getHeartTexture** () const
- const std::shared_ptr< sf::Texture > & **getKominLeft** () const
- const std::shared_ptr< sf::Texture > & **getKominLeftTop** () const
- const std::shared_ptr< sf::Texture > & **getKominRight** () const
- const std::shared_ptr< sf::Texture > & **getKominRightTop** () const

- const std::shared_ptr< sf::Texture > & **getEmptyBrickTexture** () const
- const std::shared_ptr< sf::Texture > & **getBackgroundTexture** () const
- void loadTexturesToStorage ()
- const std::shared_ptr< sf::Texture > & **getBrickTexture** () const
- **TexturesStorage** (TexturesStorage &other)=delete
- void **operator=** (const TexturesStorage &)=delete
- const std::shared_ptr< sf::Texture > & **getSoilTexture** () const
- const std::shared_ptr< sf::Texture > & **getGumbaTexture** () const
- const std::shared_ptr< sf::Texture > & **getTurtleWalkingTexture** () const
- const std::shared_ptr< sf::Texture > & **getTurtleRunningTexture** () const
- const std::shared_ptr< sf::Texture > & **getPatykTop** () const
- const std::shared_ptr< sf::Texture > & **getPatyk** () const
- const std::shared_ptr< sf::Texture > & **getKrzakLeft** () const
- const std::shared_ptr< sf::Texture > & **getKrzak** () const
- const std::shared_ptr< sf::Texture > & **getKrzakRight** () const
- const std::shared_ptr< sf::Texture > & **getChmuraTopLeft** () const
- const std::shared_ptr< sf::Texture > & **getChmuraTop** () const
- const std::shared_ptr< sf::Texture > & **getChmuraTopRight** () const
- const std::shared_ptr< sf::Texture > & **getChmuraBottomLeft** () const
- const std::shared_ptr< sf::Texture > & **getChmuraBottom** () const
- const std::shared_ptr< sf::Texture > & **getChmuraBottomRight** () const

## Statyczne metody publiczne

- static std::shared_ptr< TexturesStorage > **getInstance** ()

## 3.17.1 Dokumentacja funkcji składowych

### 3.17.1.1 loadTexturesToStorage()

```
void TexturesStorage::loadTexturesToStorage ( )
```

Wczytuje tekstury z dysku

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/TexturesStorage.h
- /home/wojciech/CLionProjects/super_mario/src/TexturesStorage.cpp

## 3.18 Dokumentacja klasy Turtle

`#include <Turtle.h>`

Diagram dziedziczenia dla Turtle

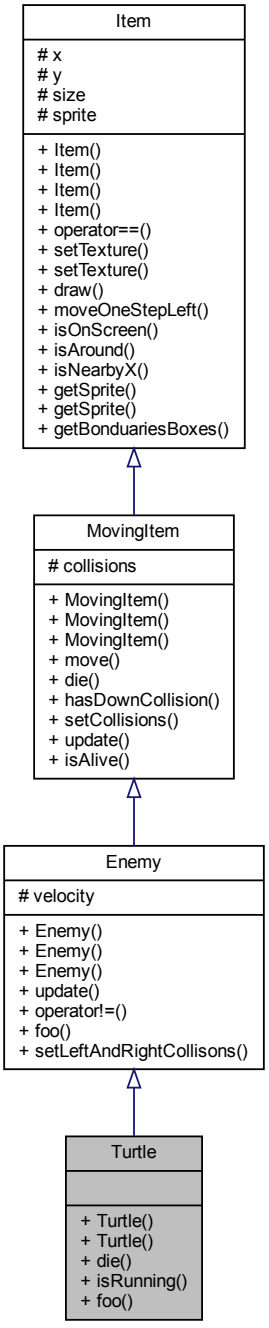
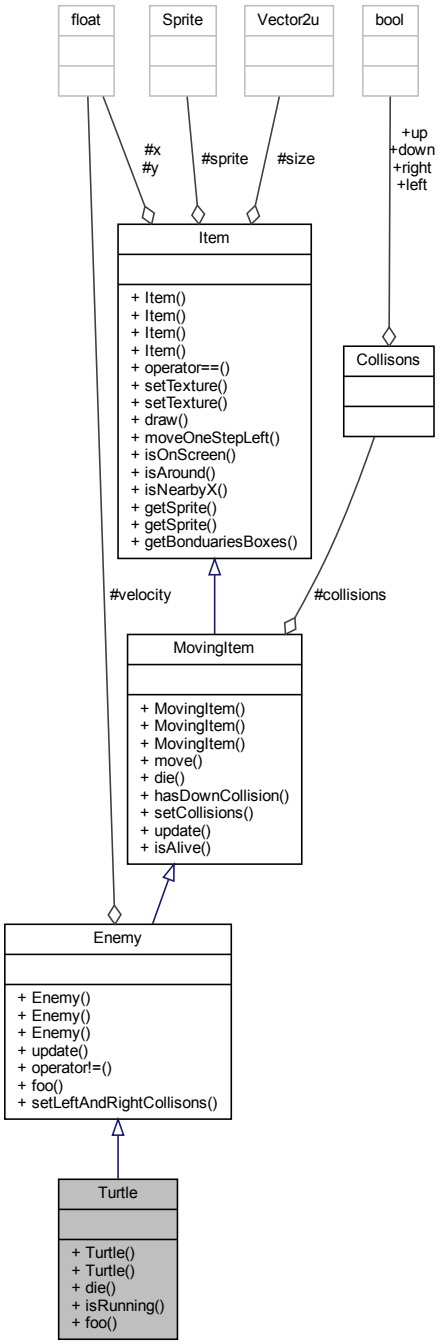


Diagram współpracy dla Turtle:




## Metody publiczne

- **Turtle** (int x, int y)
- void **die** ()
- bool **isRunning** ()
- void foo ()

### Dodatkowe Dziedziczone Składowe

### 3.18.1 Opis szczegółowy

Klasa przechowuje informacje o żółwiu

### 3.18.2 Dokumentacja funkcji składowych

#### 3.18.2.1 foo()

```
void Turtle::foo ( )  [inline], [virtual]
```

Pusta metoda definiująca niniejszą klasę jako abstrakcyjną.

Implementuje Enemy.

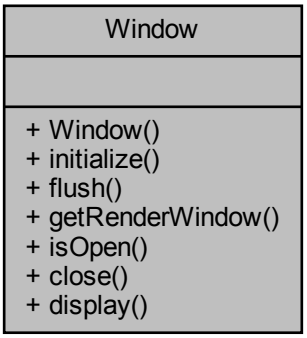
Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Turtle.h
- /home/wojciech/CLionProjects/super_mario/src/Turtle.cpp

## 3.19 Dokumentacja klasy Window

```
#include <Window.h>
```

Diagram współpracy dla Window:



## Metody publiczne

- void initialize ()
- void flush ()
- sf::RenderWindow & getRenderWindow ()
- bool isOpen ()
- void close ()
- void display ()

### 3.19.1 Opis szczegółowy

Klasa wyzwalająca i obsługująca okno

### 3.19.2 Dokumentacja funkcji składowych

#### 3.19.2.1 close()

```
void Window::close ( )  [inline]
```

Zamyka okno.

#### 3.19.2.2 display()

```
void Window::display ( )  [inline]
```

Wyświetla zawartość okna na ekranie urządzenia.

#### 3.19.2.3 flush()

```
void Window::flush ( )
```

Oczyszcza okno z zawartości.

#### 3.19.2.4 getRenderWindow()

```
sf::RenderWindow& Window::getRenderWindow ( )  [inline]
```

Getter okna.

**Zwraca**

referencja do sf::RenderWindow

#### 3.19.2.5 initialize()

```
void Window::initialize ( )
```

Nadaje parametry nowemu oknu.

#### 3.19.2.6 isOpen()

```
bool Window::isOpen ( )  [inline]
```

Getter informacji o otwarciu okna.

**Zwraca**

czy okno jest otwarte

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/wojciech/CLionProjects/super_mario/src/headers/Window.h
- /home/wojciech/CLionProjects/super_mario/src/Window.cpp

# Indeks