

Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 4

Zigiman

autor	Rafał Broncel
prowadzący	dr inż. Krzysztof Pasterak
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	4
grupa	5
sekcja	1

1 Analiza tematu

1.1 Opis projektu

Celem projektu było stworzenie gry, która jest wzorowana na popularnej grze „Pacman”. Gdzie główna postać ma za zadanie zjeść wszystkie punkty położone na mapie. Dodatkowa postać nie może być złapana przez duszki. Wyjątek od reguły stanowi sytuacja gdy zostanie zjedzony czerwony punkt (który jest odpowiednikiem dużego punktu w klasycznym pacmanie) w takim wypadku role się odwracają czyli postać sterowana przez gracza zjada duszki. W grze występują trzy rodzaje duszków:

- Guardian** porusza się tylko po wyznaczonym obszarze mapy
- Casper** porusza się po mapie
- Sprigginia** porusza się po mapie oraz ma możliwość teleportacji

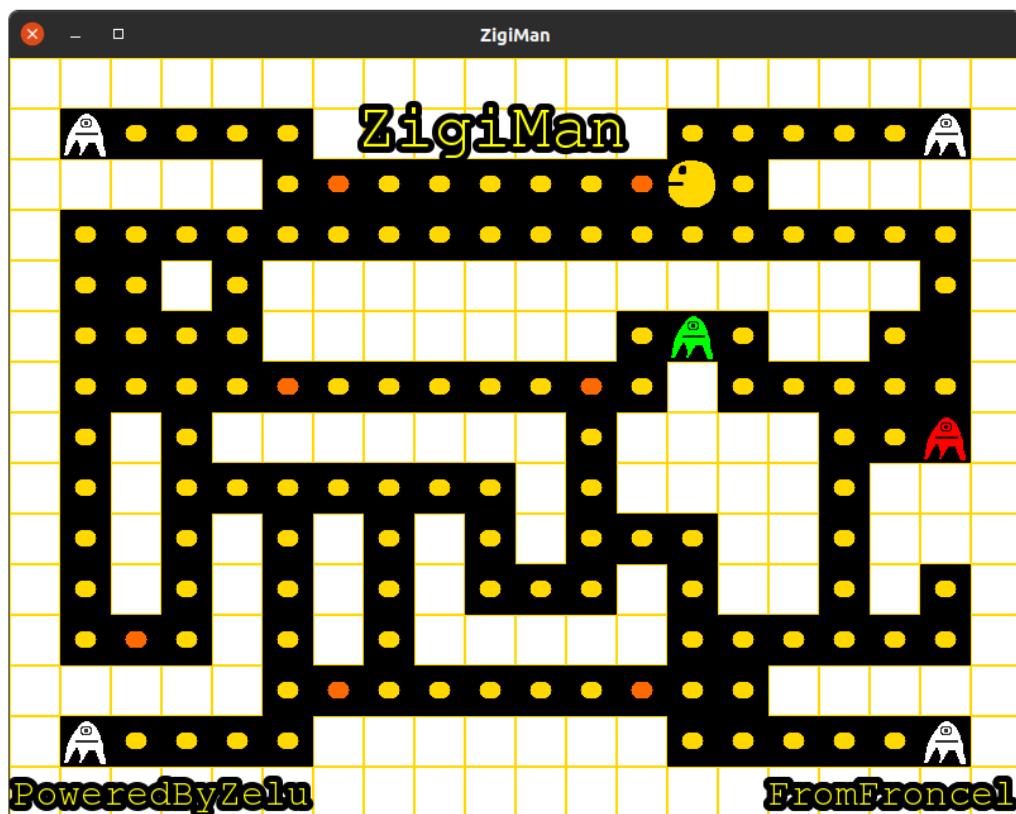
1.2 Klasy i biblioteki

W programie wykorzystałem zewnętrzną bibliotekę SFML, która znacznie ułatwiła pracę nad projektem od strony graficznej. Kod programu zawiera 14 własnoręcznie napisanych klas, szczegółowy opis wraz z hierarchią znajduje się w załączniku. Uwzględniłem następujące tematy z zajęć:

- RTTI
- mechanizmy wyjątków
- kontenery STL
- algorytmy i iteratory STL
- inteligentne wskaźniki

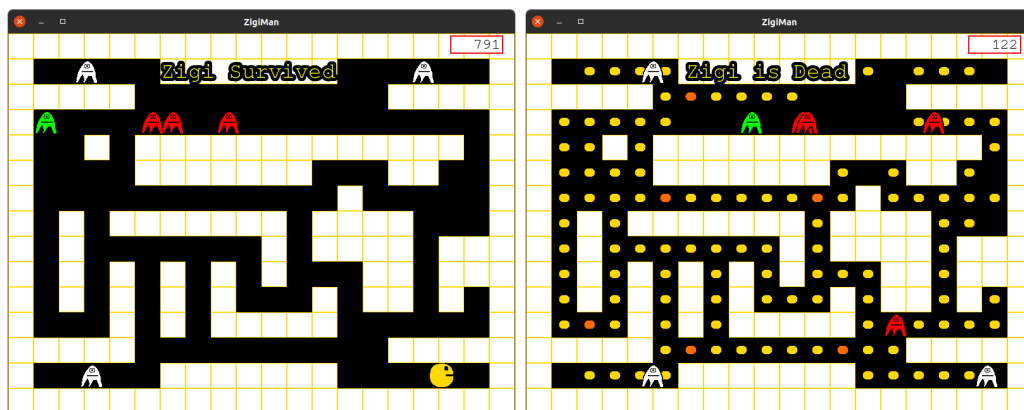
2 Specyfikacja zewnętrzna

Po uruchomieniu gry pojawia się poniższe okno:



Aby zagrać wystarczy poruszyć postać za pomocą strzałek!

W przypadku gdy gracz wygrał pojawia się okno po lewej stronie, a w przypadku przegranej okno po prawej stronie:



3 Specyfikacja wewnętrzna

Gra działa w 40 klatkach na sekundę i rozdzielczości 800x600 pikseli. W komentarzach często się posługuję pojęciem **jednostka czasu** co odpowiada $1/40$ sekundy. Program był tworzony za pomocą kompilatora **GCC** pod systemem **Ubuntu 20.04 LTS (Focal Fossa)**. Szczegółowy opis wszystkich klas, struktur i metod znajduje się w załączniku.

4 Testowanie i Uruchamianie

Największym problem okazały się kolizje z ścianą. Oczywiście wiedziałem, że to może być problemem jak się zetkną dwie krawędzie obiektów, ale nie spodziewałem się, że problem zabierze mi sporo czasu. Ponieważ postać w przypadku wykrycia kolizji zmieniała miejsce na mapie w sposób niepożądany (oczywiście to było kilka pikseli, ale wystarczająco by utrudniało rozgrywkę).

Próba wejścia pomiędzy dwie ściany także była nie wygodna. Dobrym rozwiązaniem okazało się zapamiętywanie poprzedniej pozycji obiektu (gracza i ducha) co klatkę. Gdy wystąpi kolizja obiekt jest przenoszony do przedniej pozycji, co za tym idzie nie ma potrzeby rozpoznawania z której strony wystąpiła kolizja, takie podejście także sprawiła, że rozgrywka jest bardzo przyjemna (oczywiście jest to subiektywna opinia).

Napotkałem, także inne problemy, takie jak znikanie punktów (żółtych i czerwonych), które polegały na tym aby punkt nie zniknął dopóki postać w całości nie „przykryje” punktu, rozwiązałem to w ten sposób, że program liczy pole powierzchni obszaru zakreślonego przez oba obiekty dzięki czemu nie muszę uwzględniać wariantu z której strony nadchodzi kolizja.

5 Wnioski

Jedyny wniosek, który mi się „nasunął” jest taki, że często rozwiązanie problemu jest proste, a sam programista chce uwzględnić miliony warunków, które mogą być nie wystarczające i niepotrzebne.

Załącznik

Szczegółowy opis klas, metod i funkcji

Zigiman

Wygenerowano przez Doxygen 1.8.17

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja klasy BigPoint	7
4.1.1 Dokumentacja funkcji składowych	7
4.1.1.1 checkCollision()	8
4.1.1.2 draw()	8
4.1.1.3 returnPointValue()	8
4.2 Dokumentacja klasy Character	9
4.2.1 Opis szczegółowy	9
4.2.2 Dokumentacja konstruktora i destruktora	10
4.2.2.1 Character()	10
4.2.3 Dokumentacja funkcji składowych	10
4.2.3.1 draw()	10
4.2.3.2 getVelocity()	10
4.2.3.3 move()	11
4.2.3.4 resolveCollison()	11
4.2.3.5 returnHitBox()	11
4.2.3.6 setDirection()	11
4.2.3.7 setPrevVelocity()	11
4.3 Dokumentacja klasy Game	12
4.3.1 Opis szczegółowy	12
4.3.2 Dokumentacja konstruktora i destruktora	12
4.3.2.1 Game()	12
4.3.3 Dokumentacja funkcji składowych	12
4.3.3.1 ChangeGhosts()	12
4.3.3.2 checkCollisionPacmanWall()	13
4.3.3.3 CollisionGhostWall()	13
4.3.3.4 collisionPacmanGhost()	13
4.3.3.5 collisionPacmanPoints()	13
4.3.3.6 collisionPacmanWall()	13
4.3.3.7 drawGhosts()	13
4.3.3.8 drawObjects()	13
4.3.3.9 drawPacman()	13
4.3.3.10 drawPoints()	14
4.3.3.11 drawWalls()	14

4.3.3.12 moveAllObjects()	14
4.3.3.13 moveGhost()	14
4.3.3.14 run()	14
4.4 Dokumentacja klasy Ghost	14
4.4.1 Opis szczegółowy	15
4.4.2 Dokumentacja konstruktora i destruktora	15
4.4.2.1 Ghost()	15
4.4.3 Dokumentacja funkcji składowych	15
4.4.3.1 angryGhostColor()	15
4.4.3.2 checkCollision()	16
4.4.3.3 draw()	16
4.4.3.4 fearGhostColor()	16
4.4.3.5 fearReaction()	16
4.4.3.6 move()	16
4.4.3.7 operator--()	17
4.4.3.8 setDead()	17
4.4.3.9 setDirection()	17
4.4.3.10 setPositionToTeleport()	17
4.4.3.11 setRandomDirection()	17
4.4.3.12 teleportTo()	17
4.5 Dokumentacja klasy GhostCasper	18
4.5.1 Opis szczegółowy	18
4.5.2 Dokumentacja funkcji składowych	18
4.5.2.1 setDirection()	18
4.6 Dokumentacja klasy GhostGuardian	19
4.6.1 Opis szczegółowy	19
4.6.2 Dokumentacja funkcji składowych	19
4.6.2.1 setDirection()	19
4.7 Dokumentacja klasy GhostSprigginia	20
4.7.1 Opis szczegółowy	20
4.7.2 Dokumentacja funkcji składowych	20
4.7.2.1 setDirection()	20
4.7.2.2 setPositionToTeleport()	20
4.7.2.3 teleportTo()	21
4.8 Dokumentacja klasy LiveObj	21
4.8.1 Opis szczegółowy	22
4.8.2 Dokumentacja konstruktora i destruktora	22
4.8.2.1 ~LiveObj()	22
4.8.3 Dokumentacja funkcji składowych	22
4.8.3.1 draw()	22
4.8.3.2 getLiveStatus()	23
4.8.3.3 setAlive()	23

4.8.3.4 setDead()	23
4.9 Dokumentacja struktury Map	23
4.9.1 Opis szczegółowy	24
4.10 Dokumentacja klasy Obj	24
4.10.1 Opis szczegółowy	24
4.10.2 Dokumentacja konstruktora i destruktora	24
4.10.2.1 Obj()	25
4.10.3 Dokumentacja funkcji składowych	26
4.10.3.1 draw()	26
4.10.3.2 getPosition()	26
4.10.3.3 setPosition()	26
4.11 Dokumentacja klasy Pacman	27
4.11.1 Opis szczegółowy	27
4.11.2 Dokumentacja konstruktora i destruktora	27
4.11.2.1 Pacman()	27
4.11.3 Dokumentacja funkcji składowych	27
4.11.3.1 draw()	27
4.11.3.2 move()	28
4.11.3.3 setDirection()	28
4.11.3.4 setRotation()	28
4.12 Dokumentacja klasy Point	28
4.12.1 Opis szczegółowy	29
4.12.2 Dokumentacja funkcji składowych	29
4.12.2.1 checkCollision()	29
4.12.2.2 draw()	29
4.13 Dokumentacja klasy PointsCounter	30
4.13.1 Opis szczegółowy	30
4.13.2 Dokumentacja konstruktora i destruktora	30
4.13.2.1 PointsCounter()	30
4.13.3 Dokumentacja funkcji składowych	30
4.13.3.1 operator++()	30
4.13.3.2 operator+=()	31
4.13.3.3 resolveSize()	31
4.13.3.4 returnValue()	31
4.13.3.5 showPoints()	31
4.14 Dokumentacja klasy SmallPoint	32
4.14.1 Opis szczegółowy	32
4.14.2 Dokumentacja funkcji składowych	32
4.14.2.1 checkCollision()	32
4.14.2.2 draw()	32
4.14.2.3 returnPointValue()	33
4.15 Dokumentacja struktury Textures	33

4.15.1 Opis szczegółowy	33
4.16 Dokumentacja klasy Wall	34
4.16.1 Dokumentacja konstruktora i destruktora	34
4.16.1.1 Wall()	34
4.16.2 Dokumentacja funkcji składowych	34
4.16.2.1 checkCollision()	34
4.16.2.2 draw()	34
5 Dokumentacja plików	37
5.1 Dokumentacja pliku header/character.h	37
5.2 Dokumentacja pliku header/game.h	37
5.2.1 Dokumentacja typów wyliczanych	38
5.2.1.1 GameStatus	38
5.3 Dokumentacja pliku header/ghost.h	38
5.4 Dokumentacja pliku header/obj.h	38
5.4.1 Dokumentacja typów wyliczanych	39
5.4.1.1 LiveStatus	39
5.5 Dokumentacja pliku header/pacman.h	39
5.6 Dokumentacja pliku header/points.h	39
5.7 Dokumentacja pliku header/wall.h	40
Indeks	41

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

Game	12
Map	23
Obj	24
LiveObj	21
Character	9
Ghost	14
GhostCasper	18
GhostSprigginia	20
GhostGuardian	19
Pacman	27
Point	28
BigPoint	7
SmallPoint	32
Wall	34
PointsCounter	30
Textures	33

Rozdział 2

Indeks klas

2.1 Lista klas

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

BigPoint	7
Character	9
Game	12
Ghost	14
GhostCasper	18
GhostGuardian	19
GhostSprigginia	20
LiveObj	21
Map	23
Obj	24
Pacman	27
Point	28
PointsCounter	30
SmallPoint	32
Textures	33
Wall	34

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

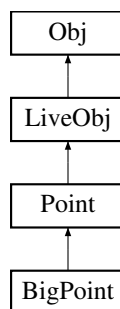
header/ character.h	37
header/ game.h	37
header/ ghost.h	38
header/ obj.h	38
header/ pacman.h	39
header/ points.h	39
header/ wall.h	40

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy BigPoint

Diagram dziedziczenia dla BigPoint



Metody publiczne

- **BigPoint** (const sf::Texture &_texture, const sf::Vector2f &_position)
- virtual void **draw** (sf::RenderWindow &_window) const
- virtual bool **checkCollision** (const std::unique_ptr< Pacman > &_pacman)
- virtual int **returnPointValue** () const

Dodatkowe Dziedziczone Składowe

4.1.1 Dokumentacja funkcji składowych

4.1.1.1 checkCollision()

```
virtual bool BigPoint::checkCollision (
    const std::unique_ptr< Pacman > & _pacman ) [virtual]
```

Zwraca

true jeśli była kolizja i zmienia status obiektu na DEAD

false jeśli nie było kolizji i nie zmienia statusu

Reimplementowana z [Point](#).

4.1.1.2 draw()

```
virtual void BigPoint::draw (
    sf::RenderWindow & _window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<i>[in.out]</i>	_window
-----------------	---------

Reimplementowana z [Point](#).

4.1.1.3 returnPointValue()

```
virtual int BigPoint::returnPointValue ( ) const [virtual]
```

Zwraca

wartość Bigpointa

Implementuje [Point](#).

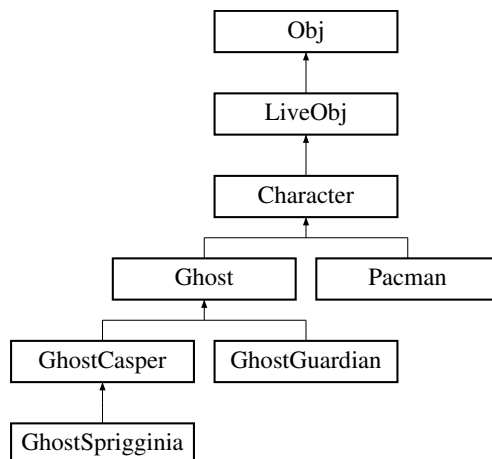
Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[points.h](#)

4.2 Dokumentacja klasy Character

```
#include <character.h>
```

Diagram dziedziczenia dla Character



Metody publiczne

- **Character** (const sf::Texture &_texture, const sf::Vector2f _position)
- virtual void **draw** (sf::RenderWindow &_window) const
- virtual sf::FloatRect **returnHitBox** () const
- virtual void **resolveCollison** ()
- virtual void **setDirection** ()=0
- virtual void **move** ()
- virtual sf::Vector2f **getVelocity** () const
- void **setPrevVelocity** ()

Atrybuty chronione

- sf::Vector2f **velocity**
prędkość/kierunek
- sf::Vector2f **prevVelocity**
przechowuję poprzednia prędkość/kierunek

Statyczne atrybuty chronione

- static const int **Speed** =4.0

4.2.1 Opis szczegółowy

klasa postaci bazowa dla **Ghost** i **Pacman**

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Character()

```
Character::Character (
    const sf::Texture & _texture,
    const sf::Vector2f _position )
```

konstruktor

Parametry

<code>_texture</code>	tekstura
<code>_position</code>	pozycja startowa postaci

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 draw()

```
virtual void Character::draw (
    sf::RenderWindow & _window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<code>[in.out]</code>	<code>_window</code>
-----------------------	----------------------

Reimplementowana z [LiveObj](#).

Reimplementowana w [Ghost](#) i [Pacman](#).

4.2.3.2 getVelocity()

```
virtual sf::Vector2f Character::getVelocity ( ) const [virtual]
```

Zwraca

prędkość/kierunek

4.2.3.3 move()

```
virtual void Character::move ( ) [virtual]
```

powoduje zmianę położenia obiektu

Reimplementowana w [Ghost](#) i [Pacman](#).

4.2.3.4 resolveCollison()

```
virtual void Character::resolveCollison ( ) [virtual]
```

rozwiązuje kolizje

4.2.3.5 returnHitBox()

```
virtual sf::FloatRect Character::returnHitBox ( ) const [virtual]
```

Zwraca

globalBoudns-"ramki obiektu"

4.2.3.6 setDirection()

```
virtual void Character::setDirection ( ) [pure virtual]
```

metoda czysto wirtualna

Implementowany w [Ghost](#), [GhostSprigginia](#), [GhostCasper](#), [GhostGuardian](#) i [Pacman](#).

4.2.3.7 setPrevVelocity()

```
void Character::setPrevVelocity ( )
```

ustawia poprzedni kierunek/prędkość

Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[character.h](#)

4.3 Dokumentacja klasy Game

```
#include <game.h>
```

Metody publiczne

- [Game](#) ()
- void [run](#) ()
- void [drawObjects](#) ()
- void [drawWalls](#) ()
- void [drawPoints](#) ()
- void [drawPacman](#) ()
- void [drawGhosts](#) ()
- void [ChangeGhosts](#) ()
- void [collisionPacmanPoints](#) ()
- bool [collisionPacmanGhost](#) ()
- bool [checkCollisionPacmanWall](#) ()
- void [collisionPacmanWall](#) ()
- bool [CollisionGhostWall](#) ()
- void [moveAllObjects](#) ()
- void [moveGhost](#) ()

4.3.1 Opis szczegółowy

klasa gry w której są wywoływane wszystkie metody które mają wpływ na grę

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 Game()

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

konstruktor gry czyta mapę obiekty, tekstury

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 ChangeGhosts()

```
void Game::ChangeGhosts ( )
```

odpowada za miganie duszków jeśli czas super mocy pacmana się kończy

4.3.3.2 checkCollisionPacmanWall()

```
bool Game::checkCollisionPacmanWall ( )
```

sprawdza kolizje Pacmana z ścianą

4.3.3.3 CollisionGhostWall()

```
bool Game::CollisionGhostWall ( )
```

metoda wykrywa i rozwiązuje kolizje duszków z ścianami

4.3.3.4 collisionPacmanGhost()

```
bool Game::collisionPacmanGhost ( )
```

metoda wykrywa i odpowiedni reaguje na kolizje pacmana z duszkami

4.3.3.5 collisionPacmanPoints()

```
void Game::collisionPacmanPoints ( )
```

metoda wykrywa i rozwiązuje kolizje pacmana z punktami

4.3.3.6 collisionPacmanWall()

```
void Game::collisionPacmanWall ( )
```

metoda wykrywa i rozwiązuje kolizje pacmana z ścianą

4.3.3.7 drawGhosts()

```
void Game::drawGhosts ( )
```

rysuje wszystkie duchy w grze

4.3.3.8 drawObjects()

```
void Game::drawObjects ( )
```

rysuje wszystkie obiekty w grze

4.3.3.9 drawPacman()

```
void Game::drawPacman ( )
```

rysuje Pacmana

4.3.3.10 drawPoints()

```
void Game::drawPoints ( )
```

rysuje wszystkie punkty (małe i duże) w grze

4.3.3.11 drawWalls()

```
void Game::drawWalls ( )
```

rysuje wszystkie ściany

4.3.3.12 moveAllObjects()

```
void Game::moveAllObjects ( )
```

przesuwa i ustawia wszystkie obiekty

4.3.3.13 moveGhost()

```
void Game::moveGhost ( )
```

przesuwa wszystkie duszki

4.3.3.14 run()

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

główna metoda gry w niej dzieje się cała akcja

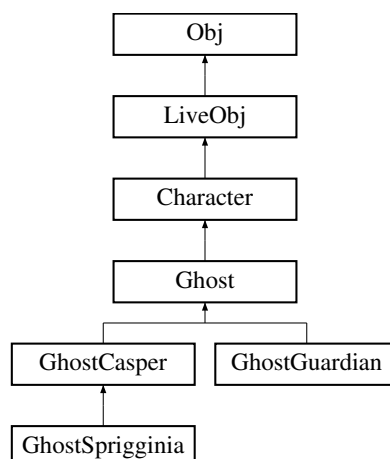
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [header/game.h](#)

4.4 Dokumentacja klasy Ghost

```
#include <ghost.h>
```

Diagram dziedziczenia dla Ghost



Metody publiczne

- `Ghost` (const sf::Texture &_texture, const sf::Vector2f &_position, sf::Color _color)
- void `fearGhostColor` ()
- void `angryGhostColor` ()
- virtual void `draw` (sf::RenderWindow &_window) const
- virtual bool `checkCollision` (const std::unique_ptr< `Pacman` > &_pacman)
- virtual void `move` ()
- virtual void `setDirection` ()=0
- virtual void `setPositionToTeleport` (const sf::Vector2f &_position)
- virtual void `teleportTo` (const sf::Vector2f &_velocity)
- int `operator--` (int)
- void `setDead` ()
- void `fearReaction` ()
- int `returnGhostvalue` ()

Metody chronione

- void `setRandomDirection` ()

Dodatkowe Dziedziczone Składowe

4.4.1 Opis szczegółowy

klasa abstrkcyjna ducha

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 Ghost()

```
Ghost::Ghost (
    const sf::Texture & _texture,
    const sf::Vector2f &_position,
    sf::Color _color )
```

konstruktor ustawia pozycję, teksturę i kolor duszka

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 angryGhostColor()

```
void Ghost::angryGhostColor ( )
```

ustawia kolor duszka gdy pacman ucieka

4.4.3.2 checkCollision()

```
virtual bool Ghost::checkCollision (
    const std::unique_ptr< Pacman > & _pacman ) [virtual]
```

sprawdza kolizje z pacmanem

Zwraca

true jesli nastąpiła kolizja

false jesli kolizja nie wystąpiła

4.4.3.3 draw()

```
virtual void Ghost::draw (
    sf::RenderWindow & _window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<i>[in.out]</i>	_window
-----------------	---------

Reimplementowana z [Character](#).

4.4.3.4 fearGhostColor()

```
void Ghost::fearGhostColor ( )
```

ustawia kolor duszka gdy pacman walczy

4.4.3.5 fearReaction()

```
void Ghost::fearReaction ( )
```

reakcja duszków na zjedzenie BigPointa przez pacmana

4.4.3.6 move()

```
virtual void Ghost::move ( ) [virtual]
```

przesuwa duszka

Reimplementowana z [Character](#).

4.4.3.7 operator--()

```
int Ghost::operator-- (
    int )
```

dekrementuje czas, kiedy duszek jest martwy

4.4.3.8 setDead()

```
void Ghost::setDead ( )
```

ustawia status na DEAD i ustawia czas bycia martwym

4.4.3.9 setDirection()

```
virtual void Ghost::setDirection ( ) [pure virtual]
```

metoda czysto wirtualna

Implementuje [Character](#).

Implementowany w [GhostSprigginia](#), [GhostCasper](#) i [GhostGuardian](#).

4.4.3.10 setPositionToTeleport()

```
virtual void Ghost::setPositionToTeleport (
    const sf::Vector2f & _position ) [inline], [virtual]
```

metoda konieczna aby zachować hierarchię klas

Reimplementowana w [GhostSprigginia](#).

4.4.3.11 setRandomDirection()

```
void Ghost::setRandomDirection ( ) [protected]
```

metoda losuje kierunek poruszania się

4.4.3.12 teleportTo()

```
virtual void Ghost::teleportTo (
    const sf::Vector2f & _velocity ) [inline], [virtual]
```

metoda konieczna aby zachować hierarchię klas

Reimplementowana w [GhostSprigginia](#).

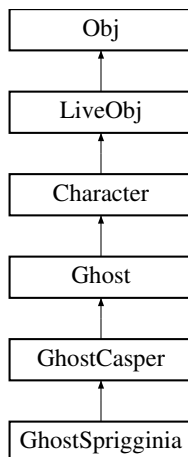
Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[ghost.h](#)

4.5 Dokumentacja klasy GhostCasper

```
#include <ghost.h>
```

Diagram dziedziczenia dla GhostCasper



Metody publiczne

- **GhostCasper** (const sf::Texture &_texture, const sf::Vector2f _position, sf::Color _color)
- virtual void [setDirection](#) ()

Dodatkowe Dziedziczone Składowe

4.5.1 Opis szczegółowy

klasyczny duszek który się porusza po mapie

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 setDirection()

```
virtual void GhostCasper::setDirection ( ) [virtual]
```

ustawia kierunek ruchu

Implementuje [Ghost](#).

Reimplementowana w [GhostSprigginia](#).

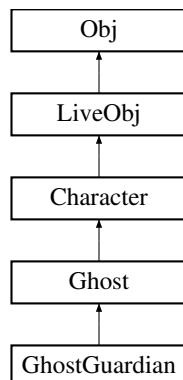
Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[ghost.h](#)

4.6 Dokumentacja klasy GhostGuardian

```
#include <ghost.h>
```

Diagram dziedziczenia dla GhostGuardian



Metody publiczne

- **GhostGuardian** (const sf::Texture &_texture, const sf::Vector2f _position, sf::Color _color)
- virtual void [setDirection](#) ()

Dodatkowe Dziedziczone Składowe

4.6.1 Opis szczegółowy

duszek który patroluje w kółko określony teren

4.6.2 Dokumentacja funkcji składowych

4.6.2.1 setDirection()

```
virtual void GhostGuardian::setDirection ( ) [virtual]
```

ustawia kierunek ruchu zależnie od kolizja tylko w prawa bądź w lewo

Implementuje [Ghost](#).

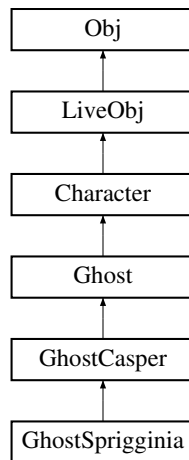
Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[ghost.h](#)

4.7 Dokumentacja klasy GhostSprigginia

```
#include <ghost.h>
```

Diagram dziedziczenia dla GhostSprigginia



Metody publiczne

- **GhostSprigginia** (sf::Texture &_texture, const sf::Vector2f &_position, sf::Color _color)
- virtual void [setDirection](#) ()
- virtual void [setPositionToTeleport](#) (const sf::Vector2f &_position)
- virtual void [teleportTo](#) (const sf::Vector2f &_velocity)

Dodatkowe Dziedziczone Składowe

4.7.1 Opis szczegółowy

klasa duszka Sprigginia duszek posiada moc teleportowania sie w miejsce które odwiedził pacman pod koniec czasu trwania supermocy (czyli gdy zjadł BigPointa) pacmana

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 setDirection()

```
virtual void GhostSprigginia::setDirection ( ) [virtual]
```

ustawia kierunek

Reimplementowana z [GhostCasper](#).

4.7.2.2 setPositionToTeleport()

```
virtual void GhostSprigginia::setPositionToTeleport (
    const sf::Vector2f &_position ) [virtual]
```

ustawia pozycje do teleportu

Parametry

<code>_position</code>	pozycja do której duszek będzie się teleportował
------------------------	--

Reimplementowana z [Ghost](#).

4.7.2.3 teleportTo()

```
virtual void GhostSprigginia::teleportTo (
    const sf::Vector2f & _velocity ) [virtual]
```

teleportuje duszka do wskazywanego miejsca

Parametry

<code>_velocity</code>	prędkość/kierunek jaką przyjmuje duszek po teleportacji i jest to kierunek/predkosc pacmana
------------------------	---

Reimplementowana z [Ghost](#).

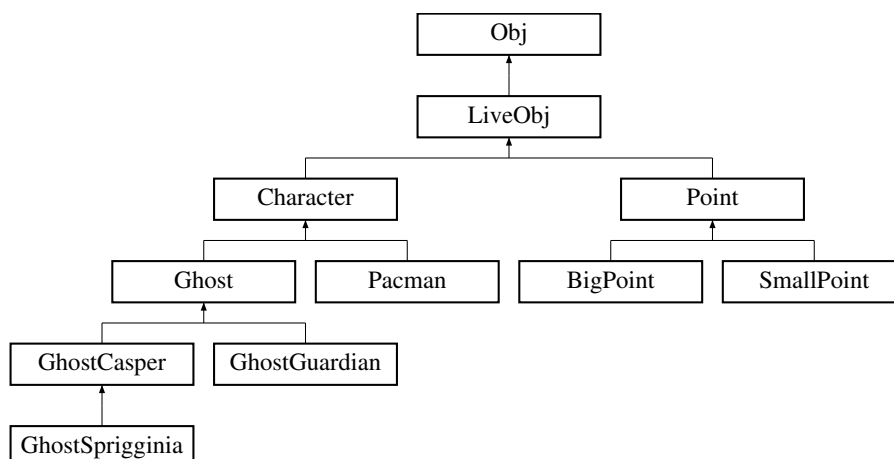
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [header/ghost.h](#)

4.8 Dokumentacja klasy LiveObj

```
#include <obj.h>
```

Diagram dziedziczenia dla LiveObj



Metody publiczne

- **LiveObj** (const sf::Texture &_texture, const sf::Vector2f _position)
- virtual void **draw** (sf::RenderWindow &_window) const
- bool **setDead** ()
- bool **setAlive** ()
- **LiveStatus** **getLiveStatus** () const
- virtual **~LiveObj** ()

Dodatkowe Dziedziczone Składowe

4.8.1 Opis szczegółowy

klasa abstrakcja podstawowa dla wszystkich obiektów które "żyją"

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 ~LiveObj()

```
virtual LiveObj::~~LiveObj ( ) [inline], [virtual]
```

destruktor

4.8.3 Dokumentacja funkcji składowych

4.8.3.1 draw()

```
virtual void LiveObj::draw (
    sf::RenderWindow & _window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<i>[in.out]</i>	_window
-----------------	---------

Reimplementowana z [Obj](#).

Reimplementowana w [BigPoint](#), [SmallPoint](#), [Character](#), [Ghost](#), [Point](#) i [Pacman](#).

4.8.3.2 getLiveStatus()

```
LiveStatus LiveObj::getLiveStatus ( ) const
```

Zwraca

status obiektu

4.8.3.3 setAlive()

```
bool LiveObj::setAlive ( )
```

ustawia status obiektu na ALIVE

Zwraca

true jeśli zmieniono status na ALIVE

false jeśli status obiektu był ALIVE

4.8.3.4 setDead()

```
bool LiveObj::setDead ( )
```

ustawia status obiektu na DEAD

Zwraca

true jeśli zmieniono status na DEAD

false jeśli status obiektu był DEAD

Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[obj.h](#)

4.9 Dokumentacja struktury Map

```
#include <game.h>
```

Atrybuty publiczne

- `std::vector< std::unique_ptr< Point > > points`
przechowuje wszystkie punkty
- `std::vector< Wall > walls`
przechowuje wszystkie sciany

4.9.1 Opis szczegółowy

struktura mapy, przechowuje sciany i punkty

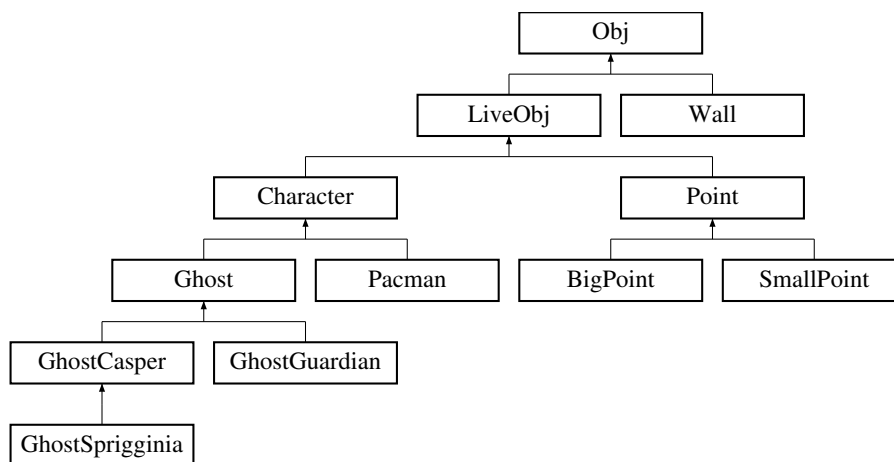
Dokumentacja dla tej struktury została wygenerowana z pliku:

- [header/game.h](#)

4.10 Dokumentacja klasy Obj

```
#include <obj.h>
```

Diagram dziedziczenia dla Obj



Metody publiczne

- [Obj](#) (const sf::Texture &_texture, const sf::Vector2f &_position)
- virtual void [draw](#) (sf::RenderWindow &_window) const
- void [setPosition](#) (const sf::Vector2f &_position)
- sf::Vector2f [getPosition](#) () const

Atrybuty chronione

- sf::Sprite [sprite](#)
reprezentant obiektu

4.10.1 Opis szczegółowy

klasa [Obj](#) jest klasą abstrakcją podstawową dla wszystkich obiektów w grze

4.10.2 Dokumentacja konstruktora i destruktor

4.10.2.1 Obj()

```
Obj::Obj (
    const sf::Texture & _texture,
    const sf::Vector2f & _position )
```

konstruktor tworzy obiekt i umieszcza w wskazanej pozycji

Parametry

<code>_texture</code>	tekstura obiektu
<code>_position</code>	pozycja obiektu startowa/stała

4.10.3 Dokumentacja funkcji składowych

4.10.3.1 draw()

```
virtual void Obj::draw (
    sf::RenderWindow & _window ) const [virtual]
```

metoda rysuje sprite w oknie

Parametry

<code>in, out</code>	<code>_window</code>	okno w którym ma być rysowany sprite
----------------------	----------------------	--------------------------------------

Reimplementowana w [LiveObj](#), [BigPoint](#), [SmallPoint](#), [Character](#), [Ghost](#), [Point](#), [Pacman](#) i [Wall](#).

4.10.3.2 getPosition()

```
sf::Vector2f Obj::getPosition ( ) const
```

Zwraca

zwraca pozycje obiektu

4.10.3.3 setPosition()

```
void Obj::setPosition (
    const sf::Vector2f & _position )
```

ustawia bezwzględnie pozycje obiektu

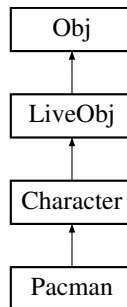
Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[obj.h](#)

4.11 Dokumentacja klasy Pacman

```
#include <pacman.h>
```

Diagram dziedziczenia dla Pacman



Metody publiczne

- `Pacman` (`const sf::Texture &_texture, const sf::Vector2f _position`)
- `virtual void draw` (`sf::RenderWindow &_window`) `const`
- `virtual void setDirection` ()
- `void setRotation` ()
- `virtual void move` ()

Dodatkowe Dziedziczone Składowe

4.11.1 Opis szczegółowy

klasa pacmana odpowiada za rysowanie, poruszanie, ustawianie kierunku i rotacji tekstury

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 Pacman()

```
Pacman::Pacman (  
    const sf::Texture &_texture,  
    const sf::Vector2f _position )
```

konstruktor ustawia pozycję i teksturę

4.11.3 Dokumentacja funkcji składowych

4.11.3.1 draw()

```
virtual void Pacman::draw (  
    sf::RenderWindow &_window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<code>[in.out]</code>	<code>_window</code>
-----------------------	----------------------

Reimplementowana z [Character](#).

4.11.3.2 move()

```
virtual void Pacman::move ( ) [virtual]
```

metoda realizuje ruch pacmana oraz zapamiętuje poprzednią pozycję

Reimplementowana z [Character](#).

4.11.3.3 setDirection()

```
virtual void Pacman::setDirection ( ) [virtual]
```

ustawia kierunek ruchu

Implementuje [Character](#).

4.11.3.4 setRotation()

```
void Pacman::setRotation ( )
```

metoda ustawia rotację pacmana zależnie od prędkości/kierunku obecnej i poprzedniej

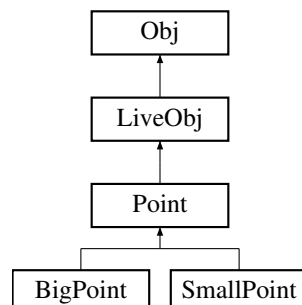
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [header/pacman.h](#)

4.12 Dokumentacja klasy Point

```
#include <points.h>
```

Diagram dziedziczenia dla Point



Metody publiczne

- **Point** (const sf::Texture &_texture, const sf::Vector2f &_position)
- virtual void **draw** (sf::RenderWindow &_window) const
- virtual bool **checkCollision** (const std::unique_ptr< **Pacman** > &_pacman)
- virtual int **returnPointValue** () const =0

Dodatkowe Dziedziczone Składowe

4.12.1 Opis szczegółowy

klasa bazowa dla punktu

4.12.2 Dokumentacja funkcji składowych

4.12.2.1 checkCollision()

```
virtual bool Point::checkCollision (  
    const std::unique_ptr< Pacman > &_pacman ) [virtual]
```

Zwraca

true jeśli była kolizja i zmienia status obiektu na DEAD
false jeśli nie było kolizji i nie zmienia statusu

Reimplementowana w **BigPoint** i **SmallPoint**.

4.12.2.2 draw()

```
virtual void Point::draw (  
    sf::RenderWindow &_window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<i>[in.out]</i>	_window
-----------------	---------

Reimplementowana z **LiveObj**.

Reimplementowana w **BigPoint** i **SmallPoint**.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [header/points.h](#)

4.13 Dokumentacja klasy PointsCounter

```
#include <game.h>
```

Metody publiczne

- [PointsCounter](#) ()
- void [resolveSize](#) ()
- void [showPoints](#) (sf::RenderWindow &_window)
- void [operator++](#) (int)
- int [operator+=](#) (int a)
- int [returnValue](#) ()

4.13.1 Opis szczegółowy

klasa licznika punktów, odpowiada za liczenie punktów i wyświetlanie licznika (w prawym górnym rogu

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 PointsCounter()

```
PointsCounter::PointsCounter ( )
```

konstruktor

4.13.3 Dokumentacja funkcji składowych

4.13.3.1 operator++()

```
void PointsCounter::operator++ (
    int )
```

przeciążony operator dodawania 1 pkt

4.13.3.2 operator+=()

```
int PointsCounter::operator+= (
    int a )
```

przeciążony operator

Zwraca

wartość po dodaniu warości

4.13.3.3 resolveSize()

```
void PointsCounter::resolveSize ( )
```

metoda odpowiednio pozycjonuje cyfry zależnie od ich liczebności

4.13.3.4 returnValue()

```
int PointsCounter::returnValue ( )
```

Zwraca

wartość licznika

4.13.3.5 showPoints()

```
void PointsCounter::showPoints (
    sf::RenderWindow & _window )
```

wyświetla punkty w prawym górnym rogu

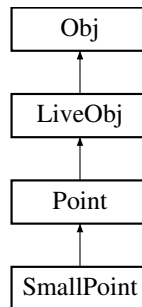
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [header/game.h](#)

4.14 Dokumentacja klasy SmallPoint

```
#include <points.h>
```

Diagram dziedziczenia dla SmallPoint



Metody publiczne

- **SmallPoint** (const sf::Texture &_texture, const sf::Vector2f &_position)
- virtual void **draw** (sf::RenderWindow &_window) const
- virtual bool **checkCollision** (const std::unique_ptr< **Pacman** > &_pacman)
- virtual int **returnPointValue** () const

Dodatkowe Dziedziczone Składowe

4.14.1 Opis szczegółowy

klasa małego punktu (na maie oznaczony kolorem żółtym)

4.14.2 Dokumentacja funkcji składowych

4.14.2.1 checkCollision()

```
virtual bool SmallPoint::checkCollision (  
    const std::unique_ptr< Pacman > &_pacman ) [virtual]
```

Zwraca

true jeśli była kolizja i zmienia status obiektu na DEAD
false jeśli nie było kolizji i nie zmienia statusu

Reimplementowana z **Point**.

4.14.2.2 draw()

```
virtual void SmallPoint::draw (  
    sf::RenderWindow &_window ) const [virtual]
```

metoda rysuje sprite w oknie jeśli jest ALIVE

Parametry

<code>[in.out]</code>	<code>_window</code>
-----------------------	----------------------

Reimplementowana z [Point](#).

4.14.2.3 returnPointValue()

```
virtual int SmallPoint::returnPointValue ( ) const [virtual]
```

Zwraca

wartość Smallpointa

Implementuje [Point](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[points.h](#)

4.15 Dokumentacja struktury Textures

```
#include <game.h>
```

Atrybuty publiczne

- sf::Texture **wall**
- sf::Texture **smallPoint**
- sf::Texture **bigPoint**
- sf::Texture **pacman**
- sf::Texture **ghost**

4.15.1 Opis szczegółowy

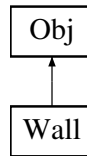
struktura przechowuje tekstury obiektów

Dokumentacja dla tej struktury została wygenerowana z pliku:

- header/[game.h](#)

4.16 Dokumentacja klasy Wall

Diagram dziedziczenia dla Wall



Metody publiczne

- [Wall](#) (const sf::Texture &_texture, const sf::Vector2f &_position)
- void [draw](#) (sf::RenderWindow &_window) const
- virtual bool [checkCollision](#) (const [Character](#) &_character)

Dodatkowe Dziedziczone Składowe

4.16.1 Dokumentacja konstruktora i destruktora

4.16.1.1 Wall()

```
Wall::Wall (
    const sf::Texture & _texture,
    const sf::Vector2f & _position )
```

kontruktor ustawiający ścianę

4.16.2 Dokumentacja funkcji składowych

4.16.2.1 checkCollision()

```
virtual bool Wall::checkCollision (
    const Character & _character ) [virtual]
```

sprawdza czy wystąpiła kolizja z ścianą

4.16.2.2 draw()

```
void Wall::draw (
    sf::RenderWindow & _window ) const [virtual]
```

metoda rysuje sprite w oknie

Parametry

<code>in, out</code>	<code>_window</code>	okno w którym ma być rysowany sprite
----------------------	----------------------	--------------------------------------

Reimplementowana z [Obj](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- header/[wall.h](#)

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku header/character.h

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include "obj.h"
#include "wall.h"
```

Komponenty

- class [Character](#)

5.2 Dokumentacja pliku header/game.h

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <memory>
#include "obj.h"
#include "character.h"
#include "ghost.h"
#include "points.h"
#include "wall.h"
#include "pacman.h"
```

Komponenty

- class [PointsCounter](#)
- struct [Map](#)
- struct [Textures](#)
- class [Game](#)

Wyliczenia

- enum `GameStatus` { `GameStatus::RUN`, `GameStatus::FIGHT` }

5.2.1 Dokumentacja typów wyliczanych

5.2.1.1 GameStatus

```
enum GameStatus [strong]
```

enum służąca do obsługi trybu gry

Wartości wyliczeń

RUN	tryb ucieczki pacmana
FIGHT	tryb walki pacmana

5.3 Dokumentacja pliku header/ghost.h

```
#include <SFML/Graphics.hpp>
#include "character.h"
#include "pacman.h"
```

Komponenty

- class `Ghost`
- class `GhostGuardian`
- class `GhostCasper`
- class `GhostSprigginia`

5.4 Dokumentacja pliku header/obj.h

```
#include <SFML/Graphics.hpp>
```

Komponenty

- class `Obj`
- class `LiveObj`

Definicje

- `#define WALL_SING '#'`
- `#define SMALL_POINT_SIGN '*'`
- `#define BIG_POINT_SIGN '&'`
- `#define PACMAN_SING 'P'`
- `#define GHOSTGUARDIAN_SING 'G'`
- `#define GHOSTCASPER_SING 'C'`
- `#define GHOSTSPRIGGINIA_SING 'S'`

Wyliczenia

- `enum LiveStatus { LiveStatus::DEAD, LiveStatus::ALIVE }`

5.4.1 Dokumentacja typów wyliczanych

5.4.1.1 LiveStatus

```
enum LiveStatus [strong]
```

enum reprezentujący status obiektu

Wartości wyliczeń

DEAD	gdzy obiekt zniszczony/zjedzony itp
ALIVE	gdzy obiekt jest wyświetlany

5.5 Dokumentacja pliku header/pacman.h

```
#include <SFML/Graphics.hpp>
#include "character.h"
```

Komponenty

- `class Pacman`

5.6 Dokumentacja pliku header/points.h

```
#include <SFML/Graphics.hpp>
#include "obj.h"
#include "character.h"
#include "pacman.h"
```

Komponenty

- class [Point](#)
- class [SmallPoint](#)
- class [BigPoint](#)

5.7 Dokumentacja pliku header/wall.h

```
#include <SFML/Graphics.hpp>
#include "obj.h"
#include "character.h"
```

Komponenty

- class [Wall](#)

Indeks

~LiveObj
LiveObj, [22](#)

ALIVE
obj.h, [39](#)
angryGhostColor
Ghost, [15](#)

BigPoint, [7](#)
checkCollision, [7](#)
draw, [8](#)
returnPointValue, [8](#)

ChangeGhosts
Game, [12](#)

Character, [9](#)
Character, [10](#)
draw, [10](#)
getVelocity, [10](#)
move, [10](#)
resolveCollison, [11](#)
returnHitBox, [11](#)
setDirection, [11](#)
setPrevVelocity, [11](#)

checkCollision
BigPoint, [7](#)
Ghost, [15](#)
Point, [29](#)
SmallPoint, [32](#)
Wall, [34](#)

checkCollisionPacmanWall
Game, [12](#)

CollisionGhostWall
Game, [13](#)

collisionPacmanGhost
Game, [13](#)

collisionPacmanPoints
Game, [13](#)

collisionPacmanWall
Game, [13](#)

DEAD
obj.h, [39](#)

draw
BigPoint, [8](#)
Character, [10](#)
Ghost, [16](#)
LiveObj, [22](#)
Obj, [26](#)
Pacman, [27](#)

Point, [29](#)
SmallPoint, [32](#)
Wall, [34](#)

drawGhosts
Game, [13](#)

drawObjects
Game, [13](#)

drawPacman
Game, [13](#)

drawPoints
Game, [13](#)

drawWalls
Game, [14](#)

fearGhostColor
Ghost, [16](#)

fearReaction
Ghost, [16](#)

FIGHT
game.h, [38](#)

Game, [12](#)
ChangeGhosts, [12](#)
checkCollisionPacmanWall, [12](#)
CollisionGhostWall, [13](#)
collisionPacmanGhost, [13](#)
collisionPacmanPoints, [13](#)
collisionPacmanWall, [13](#)
drawGhosts, [13](#)
drawObjects, [13](#)
drawPacman, [13](#)
drawPoints, [13](#)
drawWalls, [14](#)
Game, [12](#)
moveAllObjects, [14](#)
moveGhost, [14](#)
run, [14](#)

game.h
FIGHT, [38](#)
GameStatus, [38](#)
RUN, [38](#)

GameStatus
game.h, [38](#)

getLiveStatus
LiveObj, [22](#)

getPosition
Obj, [26](#)

getVelocity
Character, [10](#)

Ghost, [14](#)

- angryGhostColor, 15
- checkCollision, 15
- draw, 16
- fearGhostColor, 16
- fearReaction, 16
- Ghost, 15
- move, 16
- operator--, 16
- setDead, 17
- setDirection, 17
- setPositionToTeleport, 17
- setRandomDirection, 17
- teleportTo, 17
- GhostCasper, 18
 - setDirection, 18
- GhostGuardian, 19
 - setDirection, 19
- GhostSprigginia, 20
 - setDirection, 20
 - setPositionToTeleport, 20
 - teleportTo, 21
- header/character.h, 37
- header/game.h, 37
- header/ghost.h, 38
- header/obj.h, 38
- header/pacman.h, 39
- header/points.h, 39
- header/wall.h, 40
- LiveObj, 21
 - ~LiveObj, 22
 - draw, 22
 - getLiveStatus, 22
 - setAlive, 23
 - setDead, 23
- LiveStatus
 - obj.h, 39
- Map, 23
- move
 - Character, 10
 - Ghost, 16
 - Pacman, 28
- moveAllObjects
 - Game, 14
- moveGhost
 - Game, 14
- Obj, 24
 - draw, 26
 - getPosition, 26
 - Obj, 24
 - setPosition, 26
- obj.h
 - ALIVE, 39
 - DEAD, 39
 - LiveStatus, 39
- operator++
 - PointsCounter, 30
- operator+=
 - PointsCounter, 30
- operator--
 - Ghost, 16
- Pacman, 27
 - draw, 27
 - move, 28
 - Pacman, 27
 - setDirection, 28
 - setRotation, 28
- Point, 28
 - checkCollision, 29
 - draw, 29
- PointsCounter, 30
 - operator++, 30
 - operator+=, 30
 - PointsCounter, 30
 - resolveSize, 31
 - returnValue, 31
 - showPoints, 31
- resolveCollison
 - Character, 11
- resolveSize
 - PointsCounter, 31
- returnHitBox
 - Character, 11
- returnPointValue
 - BigPoint, 8
 - SmallPoint, 33
- returnValue
 - PointsCounter, 31
- RUN
 - game.h, 38
- run
 - Game, 14
- setAlive
 - LiveObj, 23
- setDead
 - Ghost, 17
 - LiveObj, 23
- setDirection
 - Character, 11
 - Ghost, 17
 - GhostCasper, 18
 - GhostGuardian, 19
 - GhostSprigginia, 20
 - Pacman, 28
- setPosition
 - Obj, 26
- setPositionToTeleport
 - Ghost, 17
 - GhostSprigginia, 20
- setPrevVelocity
 - Character, 11
- setRandomDirection

- Ghost, [17](#)
- setRotation
 - Pacman, [28](#)
- showPoints
 - PointsCounter, [31](#)
- SmallPoint, [32](#)
 - checkCollision, [32](#)
 - draw, [32](#)
 - returnPointValue, [33](#)
- teleportTo
 - Ghost, [17](#)
 - GhostSprigginia, [21](#)
- Textures, [33](#)
- Wall, [34](#)
 - checkCollision, [34](#)
 - draw, [34](#)
 - Wall, [34](#)