Worms 2D

1.0

Wygenerowano przez Doxygen 1.8.14

Spis treści

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

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 4

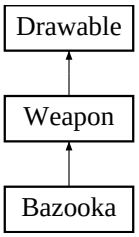
Dokumentacja klas

4.1 Dokumentacja klasy Bazooka

Klasa bazooki.

`#include <Bazooka.h>`

Diagram dziedziczenia dla Bazooka



Metody publiczne

- Bazooka ()=delete

  Konstruktor domyślny który jest usunięty.
- Bazooka (float x, float y)

  Konstruktor broni.
- virtual ~Bazooka ()=default

  Domyślny destruktor.
- void playShootSound () override

  Przeciążona funkcja odtwarzająca odgłos wystrzału z broni.

Dodatkowe Dziedziczone Składowe

4.1.1 Opis szczegółowy

Klasa bazooki.

Definicja w linii 7 pliku Bazooka.h.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 Bazooka() [1/2]

```
Bazooka::Bazooka ( )  [delete]
```

Konstruktor domyślny który jest usunięty.

4.1.2.2 Bazooka() [2/2]

```
Bazooka::Bazooka (
            float x,
            float y )
```

Konstruktor broni.

Parametry

x	- współrzędna x wyświetlenia miejsca broni
y	- współrzędna y wyświetlenia miejsca broni

Definicja w linii 4 pliku Bazooka.cpp.

4.1.2.3 ~Bazooka()

```
virtual Bazooka::~Bazooka ( )  [virtual], [default]
```

Domyślny destruktor.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 playShootSound()

```
void Bazooka::playShootSound ( )  [override], [virtual]
```

Przeciążona funkcja odtwarzająca odgłos wystrzału z broni.

Reimplementowana z Weapon.

Definicja w linii 17 pliku Bazooka.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

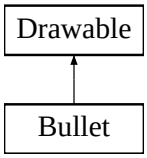
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bazooka.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bazooka.cpp

4.2 Dokumentacja klasy Bullet

Klasa pocisku.

`#include <Bullet.h>`

Diagram dziedziczenia dla Bullet



Metody publiczne

- Bullet ()=delete

  Konstruktor domyślny który jest usunięty.
- Bullet (float x, float y)

  konstruktor pocisku
- virtual ~Bullet ()=default

  Destruktor domyślny.
- void setScale (float scale)

  Ustawia skale pocisku.
- void setRotation (float rotation)

  Ustawia rotacje pocisku.
- sf::Sprite getSprite () const

  Zwraca obiekt wyświetlającego się pocisku.
- void setVelocity (sf::Vector2f velocity)

  Ustawia wektor prędkości pocisku.
- void setVelocity (float x, float y=0)

  Ustawia wektor prędkości pocisku.
- sf::Vector2f getVelocity () const

  Zwraca wektor prędkości pocisku.
- float getPosX () const

  Zwraca X pozycji pocisku na ekranie.
- float getPosY () const

  Zwraca Y pozycji pocisku na ekranie.
- void setPosX (float x)

  Ustawia wartość X pozycji pocisku na ekranie.
- void setPosY (float y)

  Ustawia wartość Y pozycji pocisku na ekranie.
- void update ()

  Aktualizuje obiekt co klatke.
- float getScale () const

  Zwraca wartość skali pocisku.
- void setScaleVector (sf::Vector2f scale)

  Ustawia wektor skali pocisku.
- sf::Vector2f getScaleVector () const

  Zwraca wektor skali pocisku.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Rysuje obiekt na ekranie (dzidziczona z SFML)

4.2.1 Opis szczegółowy

Klasa pocisku.

Definicja w linii 10 pliku Bullet.h.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Bullet() [1/2]

```
Bullet::Bullet ( )  [delete]
```

Konstruktor domyślny który jest usunięty.

4.2.2.2 Bullet() [2/2]

```
Bullet::Bullet (
            float x,
            float y )
```

konstruktor pocisku

Parametry

x	- współrzędna X pojawienia się pocisku na ekranie
y	- współrzędna Y pojawienia się pocisku na ekranie

Definicja w linii 5 pliku Bullet.cpp.

4.2.2.3 ~Bullet()

```
virtual Bullet::~Bullet ( )  [virtual], [default]
```

Destruktor domyślny.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 draw()

```
void Bullet::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 94 pliku Bullet.cpp.

4.2.3.2 getPosX()

```
float Bullet::getPosX ( ) const
```

Zwraca X pozycji pocisku na ekranie.

Zwraca

X pozycji pocisku na ekranie

Definicja w linii 49 pliku Bullet.cpp.

4.2.3.3 getPosY()

```
float Bullet::getPosY ( ) const
```

Zwraca Y pozycji pocisku na ekranie.

Zwraca

Y pozycji pocisku na ekranie

Definicja w linii 54 pliku Bullet.cpp.

4.2.3.4 getScale()

```
float Bullet::getScale ( ) const
```

Zwraca wartość skali pocisku.

Zwraca

wartość skali pocisku

Definicja w linii 78 pliku Bullet.cpp.

4.2.3.5 getScaleVector()

```
sf::Vector2f Bullet::getScaleVector ( ) const
```

Zwraca wektor skali pocisku.

Zwraca

wektor skali pocisku

Definicja w linii 88 pliku Bullet.cpp.

4.2.3.6 getSprite()

```
sf::Sprite Bullet::getSprite ( ) const
```

Zwraca obiekt wyświetlającego się pocisku.

Zwraca

obiekt sprite'a pocisku

Definicja w linii 28 pliku Bullet.cpp.

4.2.3.7 getVelocity()

```
sf::Vector2f Bullet::getVelocity ( ) const
```

Zwraca wektor prędkości pocisku.

Zwraca

wektor prędkości pocisku

Definicja w linii 44 pliku Bullet.cpp.

4.2.3.8 setPosX()

```
void Bullet::setPosX (
            float x )
```

Ustawia wartość X pozycji pocisku na ekranie.

Parametry

x	- współrzędna X pocisku na ekranie

Definicja w linii 59 pliku Bullet.cpp.

4.2.3.9 setPosY()

```
void Bullet::setPosY (
            float y )
```

Ustawia wartość Y pozycji pocisku na ekranie.

Parametry

y	- współrzędna Y pocisku na ekranie

Definicja w linii 64 pliku Bullet.cpp.

4.2.3.10 setRotation()

```
void Bullet::setRotation (
            float rotation )
```

Ustawia rotacje pocisku.

Parametry

rotation	- rotacja pocisku

Definicja w linii 23 pliku Bullet.cpp.

4.2.3.11 setScale()

```
void Bullet::setScale (
            float scale )
```

Ustawia skale pocisku.

Parametry

scale	- skala pocisku

Definicja w linii 18 pliku Bullet.cpp.

4.2.3.12 setScaleVector()

```
void Bullet::setScaleVector (
            sf::Vector2f scale )
```

Ustawia wektor skali pocisku.

Parametry

scale	- wektor skali pocisku

Definicja w linii 83 pliku Bullet.cpp.

4.2.3.13 setVelocity() [1/2]

```
void Bullet::setVelocity (
            sf::Vector2f velocity )
```

Ustawia wektor prędkości pocisku.

Parametry

velocity	- wektor prędkości pocisku

Definicja w linii 33 pliku Bullet.cpp.

4.2.3.14 setVelocity() [2/2]

```
void Bullet::setVelocity (
            float x,
            float y = 0 )
```

Ustawia wektor prędkości pocisku.

Parametry

x	- składowa X wektora prędkości pocisku
y	- składowa Y wektora prędkości pocisku

Definicja w linii 38 pliku Bullet.cpp.

4.2.3.15 update()

```
void Bullet::update ( )
```

Aktualizuje obiekt co klatke.

Definicja w linii 69 pliku Bullet.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

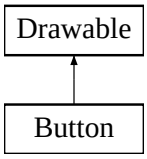
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bullet.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bullet.cpp

4.3 Dokumentacja klasy Button

Klasa przycisku.

```
#include <Button.h>
```

Diagram dziedziczenia dla Button



Metody publiczne

- Button (float position_x, float position_y, const char *label, int id)
  konstruktor przycisku
- sf::Vector2f getSize ()
  Zwraca rozmiar buttona.
- sf::Vector2f getPosition ()
  Zwraca wektor pozycji buttona.
- void hover ()
  Podswietla przycisk.
- void click ()
  Obsuguje wcinie przycisku.
- void clear ()
  Czyci przycisk.
- bool isHovered ()
  Sprawdza czy myszka jest na przycisku.
- bool isClicked ()
  Sprawdza czy przycisk zosta kliniety.
- void setColorDefault ()
  Ustawia kolor domylny przycisku.
- void setColorHovered ()
  Ustawia kolor podwietlenia przycisku.
- void setColorClicked ()
  Ustawia kolor kliknigo przycisku.
- int getId ()
  Zwraca ID przycisku.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Rysuje obiekt na ekranie (dzidziczona z SFML)

4.3.1 Opis szczegółowy

Klasa przycisku.

Definicja w linii 7 pliku Button.h.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 Button()

```
Button::Button (
            float position_x,
            float position_y,
            const char * label,
            int id )
```

konstruktor przycisku

Parametry

position_x	- pozycja X przycisku na ekranie
position_y	- pozycja Y przycisku na ekranie
label	- tekst na przycisku
id	- id przycisku

Definicja w linii 4 pliku Button.cpp.

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 clear()

```
void Button::clear ( )
```

Czyci przycisk.

Definicja w linii 60 pliku Button.cpp.

4.3.3.2 click()

```
void Button::click ( )
```

Obsuguje wcinie przycisku.

Definicja w linii 54 pliku Button.cpp.

4.3.3.3 draw()

```
void Button::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 31 pliku Button.cpp.

4.3.3.4 getId()

```
int Button::getId ( )
```

Zwraca ID przycisku.

Zwraca

ID przycisku

Definicja w linii 92 pliku Button.cpp.

4.3.3.5 getPosition()

```
sf::Vector2f Button::getPosition ( )
```

Zwraca wektor pozycji buttona.

Zwraca

wektor pozycji buttona

Definicja w linii 43 pliku Button.cpp.

4.3.3.6 getSize()

```
sf::Vector2f Button::getSize ( )
```

Zwraca rozmiar buttona.

Zwraca

wektor rozmiaru buttona

Definicja w linii 38 pliku Button.cpp.

4.3.3.7 hover()

```
void Button::hover ( )
```

Podswietla przycisk.

Definicja w linii 48 pliku Button.cpp.

4.3.3.8 isClicked()

```
bool Button::isClicked ( )
```

Sprawdza czy przycisk zosta kliniety.

Zwraca

czy przycisk zosta kliniety

Definicja w linii 72 pliku Button.cpp.

4.3.3.9 isHovered()

```
bool Button::isHovered ( )
```

Sprawdza czy myszka jest na przycisku.

Zwraca

czy myszka jest na przycisku

Definicja w linii 67 pliku Button.cpp.

4.3.3.10 setColorClicked()

```
void Button::setColorClicked ( )
```

Ustawia kolor kliknigo przycisku.

Definicja w linii 87 pliku Button.cpp.

4.3.3.11 setColorDefault()

```
void Button::setColorDefault ( )
```

Ustawia kolor domylny przycisku.

Definicja w linii 77 pliku Button.cpp.

4.3.3.12 setColorHovered()

```
void Button::setColorHovered ( )
```

Ustawia kolor podwietlenia przycisku.

Definicja w linii 82 pliku Button.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Button.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Button.cpp

4.4 Dokumentacja klasy FPSCounter

Klasa licznika FPS'ów.

```
#include <FPSCounter.h>
```

Metody publiczne

- FPSCounter (unsigned int fontSize=12)
- void start ()

  Uruchamia licznik.
- void drawFPS ()

  Rysuje ilość FPS na ekranie.
- void setColor (sf::Color color)

  Ustawia kolor licznika.

4.4.1 Opis szczegółowy

Klasa licznika FPS'ów.

Definicja w linii 7 pliku FPSCounter.h.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 FPSCounter()

```
FPSCounter::FPSCounter (
            unsigned int fontSize = 12 )
```

Konstruktor licznika FPS'ów (dla DEBBUGE'a)

Parametry

fontSize	- rozmiar czcionki

Definicja w linii 6 pliku FPSCounter.cpp.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 drawFPS()

```
void FPSCounter::drawFPS ( )
```

Rysuje ilość FPS na ekranie.

Definicja w linii 34 pliku FPSCounter.cpp.

4.4.3.2 setColor()

```
void FPSCounter::setColor (
            sf::Color color )
```

Ustawia kolor licznika.

Parametry

color	- kolor licznika

Definicja w linii 42 pliku FPSCounter.cpp.

4.4.3.3 start()

```
void FPSCounter::start ( )
```

Uruchamia licznik.

Definicja w linii 20 pliku FPSCounter.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/FPSCounter.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/FPSCounter.cpp

4.5 Dokumentacja klasy GameEvent

Klasa zdarzeń gry.

```
#include <GameEvent.h>
```

Metody publiczne

- GameEvent ()
  Konstruktor zdarzeń
- virtual ~GameEvent ()
  Destruktor zdarzeń
- sf::Event GetInstance ()
  Zwraca instancje zdarzeń SFML'a.
- void handleEvents ()
  Obsługuje zdarzenia co klatke.

Statyczne metody publiczne

- static GameEvent * GetEventInstance ()
  Zwraca statyczny wskaźnik na obiekt Zdarzeń gry (Singleton)

4.5.1 Opis szczegółowy

Klasa zdarzeń gry.

Definicja w linii 9 pliku GameEvent.h.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 GameEvent()

```
GameEvent::GameEvent ( )
```

Konstruktor zdarzeń

Definicja w linii 6 pliku GameEvent.cpp.

4.5.2.2 ~GameEvent()

```
GameEvent::~GameEvent ( )  [virtual]
```

Destruktor zdarzeń

Definicja w linii 19 pliku GameEvent.cpp.

4.5.3 Dokumentacja funkcji składowych

4.5.3.1 GetEventInstance()

```
GameEvent * GameEvent::GetEventInstance ( )  [static]
```

Zwraca statyczny wskaźnik na obiekt Zdarzeń gry (Singleton)

Zwraca

wskaźnik na obiekt Zdarzeń gry

Definicja w linii 383 pliku GameEvent.cpp.

4.5.3.2 GetInstance()

```
sf::Event GameEvent::GetInstance ( )
```

Zwraca instancje zdarzeń SFML'a.

Zwraca

instancje zdarzeń SFML'a

Definicja w linii 23 pliku GameEvent.cpp.

4.5.3.3 handleEvents()

```
void GameEvent::handleEvents ( )
```

Obsługuje zdarzenia co klatke.

Definicja w linii 28 pliku GameEvent.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameEvent.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameEvent.cpp

4.6 Dokumentacja klasy GameSound

Klasa dźwięków gry.

```
#include <GameSound.h>
```

Metody publiczne

- GameSound ()
  Konstruktor.
- void PlayMainMusic ()
  Odtwarza główne audio w grze.
- void StartSample ()
  Odtwarza odgłos worma.
- void PauseSample ()
  Pauzuje odgłos worma.
- void StopSample ()
  Stopuje odgłos worma.
- void StartRevolverSound ()
  Odtwarza odgłos wystrzału rewolwera.
- void StopRevolverSound ()
  Stopuje odtwarzanie odgłosu wystrzału rewolwera.
- void StartBazookaSound ()
  Odtwarza odgłos wystrzału bazooki.
- void StopBazookaSound ()
  Stopuje odtwarzanie odgłosu wystrzału bazooki.
- void PlayDeath ()
  Odtwarza dźwięk śmierci worma.
- void StopDeath ()
  Stopuje dźwięk śmierci worma.
- void PlayWin ()
  Odtwarza dźwięk zwycięstwa.
- void StopWin ()
  Stopuje dźwięk zwycięstwa.

4.6.1 Opis szczegółowy

Klasa dźwięków gry.

Definicja w linii 9 pliku GameSound.h.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 GameSound()

```
GameSound::GameSound ( )
```

Konstruktor.

Definicja w linii 3 pliku GameSound.cpp.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 PauseSample()

```
void GameSound::PauseSample ( )
```

Pauzuje odgłos worma.

Definicja w linii 47 pliku GameSound.cpp.

4.6.3.2 PlayDeath()

```
void GameSound::PlayDeath ( )
```

Odtwarza dźwięk śmierci worma.

Definicja w linii 77 pliku GameSound.cpp.

4.6.3.3 PlayMainMusic()

```
void GameSound::PlayMainMusic ( )
```

Odtwarza główne audio w grze.

Definicja w linii 37 pliku GameSound.cpp.

4.6.3.4 PlayWin()

```
void GameSound::PlayWin ( )
```

Odtwarza dźwięk zwycięstwa.

Definicja w linii 87 pliku GameSound.cpp.

4.6.3.5 StartBazookaSound()

```
void GameSound::StartBazookaSound ( )
```

Odtwarza odgłos wystrzału bazooki.

Definicja w linii 67 pliku GameSound.cpp.

4.6.3.6 StartRevolverSound()

```
void GameSound::StartRevolverSound ( )
```

Odtwarza odgłos wystrzału rewolwera.

Definicja w linii 57 pliku GameSound.cpp.

4.6.3.7 StartSample()

```
void GameSound::StartSample ( )
```

Odtwarza odgłos worma.

Definicja w linii 42 pliku GameSound.cpp.

4.6.3.8 StopBazookaSound()

```
void GameSound::StopBazookaSound ( )
```

Stopuje odtwarzanie odgłosu wystrzału bazooki.

Definicja w linii 72 pliku GameSound.cpp.

4.6.3.9 StopDeath()

```
void GameSound::StopDeath ( )
```

Stopuje dźwięk śmierci worma.

Definicja w linii 82 pliku GameSound.cpp.

4.6.3.10 StopRevolverSound()

```
void GameSound::StopRevolverSound ( )
```

Stopuje odtwarzanie odgłosu wystrzału rewolwera.

Definicja w linii 62 pliku GameSound.cpp.

4.6.3.11 StopSample()

```
void GameSound::StopSample ( )
```

Stopuje odgłos worma.

Definicja w linii 52 pliku GameSound.cpp.

4.6.3.12 StopWin()

```
void GameSound::StopWin ( )
```

Stopuje dźwięk zwycięstwa.

Definicja w linii 92 pliku GameSound.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameSound.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameSound.cpp

4.7 Dokumentacja klasy GameWindow

Klasa okna głównego.

```
#include <GameWindow.h>
```

Metody publiczne

- GameWindow ()=delete

  konstruktor usunięty
- ∼GameWindow ()

  Destruktor.
- GameWindow (unsigned int width, unsigned int height, std::string name)

  Konstruktor tworzący okno główne.
- sf::RenderWindow ∗ GetInstance () const

  Zwraca wskaźnik na instacje okna głównego.
- void ChangeFrameLimit (unsigned int limit=60) const

  Zmienia limit klatek.
- void MainLoop ()

  Przetwarza główną pętle okna co klatke.
- void UpdateWorms (int i)

  Aktualizuje odpowiedniego worma z wektora 1 drużyny.
- void UpdateWormsB (int i)

  Aktualizuje odpowiedniego worma z wektora 2 drużyny.
- Worm ∗∗ GetCurrentWorm ()

  Zwraca podwójny wskaźnik na aktualnego worma.
- std::vector< Worm ∗ > ∗ GetWormsArray () const

  Zweraca wskaźnik na wektor 1 team'u.
- std::vector< Worm ∗ > ∗ GetWormsArrayB () const

  Zweraca wskaźnik na wektor 2 team'u.
- int GetWormCount () const

  Zwraca wielkość 1 drużyny.
- int GetWormCountB () const

  Zwraca wielkość 2 drużyny.
- int ∗ GetCurrentTeam ()

  Zwraca wskaźnik na ID aktualnego team'u.
- int ∗ GetCurrentWormID ()

  Zwraca wskaźnik na ID aktualnego worma.
- void SetBackgroundColor (sf::Color color)

  Ustawia kolor tła.
- GameSound ∗ GetGameSound () const

  Zwraca wskaźnik na obiekt dźwięków gry.
- void SwitchToRedTeam ()

  Zmienia aktualny team na Czerwony (zmiana tury)
- void SwitchToBlueTeam ()

  Zmienia aktualny team na Niebieski (zmiana tury)
- void SetChooseStates ()

  Ustawia stan aktualnego worma na wybrany a wszystkich innych na normalny.

Statyczne metody publiczne

- static GameWindow ∗ GetGameWindowInstance (unsigned int width=1280, unsigned int height=1024, std↩ ::string name="Worms 2D")

  Zwraca wskaźnik na instacje okna głównego lub tworzy je jeśli nie istniało.

Atrybuty publiczne

- Terrain terrain
  Obiekt terenu.
- Menu menu
  Obiekt Menu.
- int game_state
  Aktualny stan gry.
- bool game_started
  Informacja czy gra została wystartowana.

4.7.1 Opis szczegółowy

Klasa okna głównego.

Definicja w linii 18 pliku GameWindow.h.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 GameWindow() [1/2]

```
GameWindow::GameWindow ( )  [delete]
```

konstruktor usunięty

4.7.2.2 ~GameWindow()

```
GameWindow::~GameWindow ( )
```

Destruktor.

Definicja w linii 26 pliku GameWindow.cpp.

4.7.2.3 GameWindow() [2/2]

```
GameWindow::GameWindow (
            unsigned int width,
            unsigned int height,
            std::string name )
```

Konstruktor tworzący okno główne.

Parametry

width	- szerokość okna
height	- wysokość okna
name	- nazwa okna

Definicja w linii 4 pliku GameWindow.cpp.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 ChangeFrameLimit()

```
void GameWindow::ChangeFrameLimit (
            unsigned int limit = 60 ) const
```

Zmienia limit klatek.

Parametry

limit	- limit klatek

Definicja w linii 37 pliku GameWindow.cpp.

4.7.3.2 GetCurrentTeam()

```
int * GameWindow::GetCurrentTeam ( )
```

Zwraca wskaźnik na ID aktualnego team'u.

Zwraca

wskaźnik na ID aktualnego team'u

Definicja w linii 293 pliku GameWindow.cpp.

4.7.3.3 GetCurrentWorm()

```
Worm ** GameWindow::GetCurrentWorm ( )
```

Zwraca podwójny wskaźnik na aktualnego worma.

Zwraca

podwójny wskaźnik na aktualnego worma

Definicja w linii 268 pliku GameWindow.cpp.

4.7.3.4 GetCurrentWormID()

```
int * GameWindow::GetCurrentWormID ( )
```

Zwraca wskaźnik na ID aktualnego worma.

Zwraca

wskaźnik na ID aktualnego worma

Definicja w linii 298 pliku GameWindow.cpp.

4.7.3.5 GetGameSound()

```
GameSound * GameWindow::GetGameSound ( ) const
```

Zwraca wskaźnik na obiekt dźwięków gry.

Zwraca

wskaźnik na obiekt dźwięków gry

Definicja w linii 308 pliku GameWindow.cpp.

4.7.3.6 GetGameWindowInstance()

```
GameWindow * GameWindow::GetGameWindowInstance (
            unsigned int width = 1280,
            unsigned int height = 1024,
            std::string name = "Worms 2D" )  [static]
```

Zwraca wskaźnik na instacje okna głównego lub tworzy je jeśli nie istniało.

Zwraca

wskaźnik na instacje okna głównego

Definicja w linii 373 pliku GameWindow.cpp.

4.7.3.7 GetInstance()

```
sf::RenderWindow * GameWindow::GetInstance ( ) const
```

Zwraca wskaźnik na instacje okna głównego.

Zwraca

wskaźnik na instacje okna głównego

Definicja w linii 32 pliku GameWindow.cpp.

4.7.3.8 GetWormCount()

```
int GameWindow::GetWormCount ( ) const
```

Zwraca wielkość 1 drużyny.

Zwraca

wielkość 1 drużyny

Definicja w linii 283 pliku GameWindow.cpp.

4.7.3.9 GetWormCountB()

```
int GameWindow::GetWormCountB ( ) const
```

Zwraca wielkość 2 drużyny.

Zwraca

wielkość 2 drużyny

Definicja w linii 288 pliku GameWindow.cpp.

4.7.3.10 GetWormsArray()

```
std::vector< Worm * > * GameWindow::GetWormsArray ( ) const
```

Zweraca wskaźnik na wektor 1 team'u.

Zwraca

wskaźnik na wektor 1 team'u

Definicja w linii 273 pliku GameWindow.cpp.

4.7.3.11 GetWormsArrayB()

```
std::vector< Worm * > * GameWindow::GetWormsArrayB ( ) const
```

Zweraca wskaźnik na wektor 2 team'u.

Zwraca

wskaźnik na wektor 2 team'u

Definicja w linii 278 pliku GameWindow.cpp.

4.7.3.12 MainLoop()

```
void GameWindow::MainLoop ( )
```

Przetwarza główną pętle okna co klatke.

Definicja w linii 79 pliku GameWindow.cpp.

4.7.3.13 SetBackgroundColor()

```
void GameWindow::SetBackgroundColor (
            sf::Color color )
```

Ustawia kolor tła.

Parametry

color	- kolor tła

Definicja w linii 303 pliku GameWindow.cpp.

4.7.3.14 SetChooseStates()

```
void GameWindow::SetChooseStates ( )
```

Ustawia stan aktualnego worma na wybrany a wszystkich innych na normalny.

Definicja w linii 340 pliku GameWindow.cpp.

4.7.3.15 SwitchToBlueTeam()

```
void GameWindow::SwitchToBlueTeam ( )
```

Zmienia aktualny team na Niebieski (zmiana tury)

Definicja w linii 332 pliku GameWindow.cpp.

4.7.3.16 SwitchToRedTeam()

```
void GameWindow::SwitchToRedTeam ( )
```

Zmienia aktualny team na Czerwony (zmiana tury)

Definicja w linii 324 pliku GameWindow.cpp.

4.7.3.17 UpdateWorms()

```
void GameWindow::UpdateWorms (
            int i )
```

Aktualizuje odpowiedniego worma z wektora 1 drużyny.

Parametry

i	- numer worma w wektorze

Definicja w linii 258 pliku GameWindow.cpp.

4.7.3.18 UpdateWormsB()

```
void GameWindow::UpdateWormsB (
            int i )
```

Aktualizuje odpowiedniego worma z wektora 2 drużyny.

Parametry

i	- numer worma w wektorze

Definicja w linii 263 pliku GameWindow.cpp.

4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 game_started

```
bool GameWindow::game_started
```

Informacja czy gra została wystartowana.

Definicja w linii 147 pliku GameWindow.h.

4.7.4.2 game_state

```
int GameWindow::game_state
```

Aktualny stan gry.

Definicja w linii 142 pliku GameWindow.h.

4.7.4.3 menu

```
Menu GameWindow::menu
```

Obiekt Menu.

Definicja w linii 137 pliku GameWindow.h.

4.7.4.4 terrain

```
Terrain GameWindow::terrain
```

Obiekt terenu.

Definicja w linii 132 pliku GameWindow.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

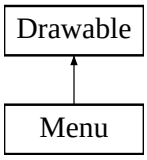
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameWindow.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameWindow.cpp

4.8 Dokumentacja klasy Menu

Klasa ggo menu.

`#include <Menu.h>`

Diagram dziedziczenia dla Menu



Metody publiczne

- Menu ()
  Konstruktor menu.
- Button ∗ getButton (float x, float y)
  Zwraca wskanik na przycisk o podanych wspych.
- void clearClickedButton ()
  Czyci klikni przycisk.
- void clear ()
  Czyci menu.
- void changeMenu (int choice)
  Zmienia menu.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override
  Rysuje obiekt na ekranie (dzidziczona z SFML)

4.8.1 Opis szczegółowy

Klasa ggo menu.

Definicja w linii 14 pliku Menu.h.

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 Menu()

`Menu::Menu ( )`

Konstruktor menu.

Definicja w linii 6 pliku Menu.cpp.

4.8.3 Dokumentacja funkcji składowych

4.8.3.1 changeMenu()

```
void Menu::changeMenu (
            int choice )
```

Zmienia menu.

Parametry

choice	- wyb

Definicja w linii 86 pliku Menu.cpp.

4.8.3.2 clear()

```
void Menu::clear ( )
```

Czyci menu.

Definicja w linii 78 pliku Menu.cpp.

4.8.3.3 clearClickedButton()

```
void Menu::clearClickedButton ( )
```

Czyci klikni przycisk.

Definicja w linii 68 pliku Menu.cpp.

4.8.3.4 draw()

```
void Menu::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 30 pliku Menu.cpp.

4.8.3.5 getButton()

```
Button * Menu::getButton (
            float x,
            float y )
```

Zwraca wskanik na przycisk o podanych wspych.

Parametry

x	- Wspa X przycisku
y	- Wspa Y przycisku

Zwraca

wskanik na przycisk

Definicja w linii 47 pliku Menu.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

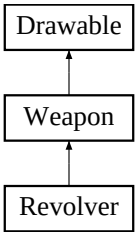
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Menu.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Menu.cpp

4.9 Dokumentacja klasy Revolver

Klasa rewolwera.

```
#include <Revolver.h>
```

Diagram dziedziczenia dla Revolver



Metody publiczne

- Revolver ()=delete

  Konstruktor domyślny który jest usunięty.
- Revolver (float x, float y)

  Konstruktor broni.
- virtual ~Revolver ()=default

  Domyślny destruktor.
- void playShootSound () override

  Przeciążona funkcja odtwarzająca odgłos wystrzału z broni.

Dodatkowe Dziedziczone Składowe

4.9.1 Opis szczegółowy

Klasa rewolwera.

Definicja w linii 7 pliku Revolver.h.

4.9.2 Dokumentacja konstruktora i destruktora

4.9.2.1 Revolver() [1/2]

```
Revolver::Revolver ( )  [delete]
```

Konstruktor domyślny który jest usunięty.

4.9.2.2 Revolver() [2/2]

```
Revolver::Revolver (
            float x,
            float y )
```

Konstruktor broni.

Parametry

x	- współrzędna x wyświetlenia miejsca broni
y	- współrzędna y wyświetlenia miejsca broni

Definicja w linii 5 pliku Revolver.cpp.

4.9.2.3 ~Revolver()

```
virtual Revolver::~Revolver ( )  [virtual], [default]
```

Domyślny destruktor.

4.9.3 Dokumentacja funkcji składowych

4.9.3.1 playShootSound()

```
void Revolver::playShootSound ( )  [override], [virtual]
```

Przeciążona funkcja odtwarzająca odgłos wystrzału z broni.

Reimplementowana z Weapon.

Definicja w linii 16 pliku Revolver.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

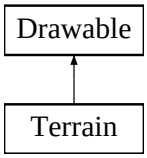
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Revolver.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Revolver.cpp

4.10 Dokumentacja klasy Terrain

Klasa terenu(mapy)

```
#include <Terrain.h>
```

Diagram dziedziczenia dla Terrain



Metody publiczne

- Terrain ()

  Konstruktor.
- void erase (const sf::Drawable &eraser)

  Usuwa element z mapy.
- void reset ()

  Resetuje mape.

Atrybuty publiczne

- sf::Image map
  Obiekt mapy.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override
  Rysuje obiekt na ekranie (dzidziczona z SFML)

4.10.1 Opis szczegółowy

Klasa terenu(mapy)

Definicja w linii 7 pliku Terrain.h.

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 Terrain()

```
Terrain::Terrain ( )
```

Konstruktor.

Definicja w linii 3 pliku Terrain.cpp.

4.10.3 Dokumentacja funkcji składowych

4.10.3.1 draw()

```
void Terrain::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 31 pliku Terrain.cpp.

4.10.3.2 erase()

```
void Terrain::erase (
            const sf::Drawable & eraser )
```

Usuwa element z mapy.

Parametry

eraser	

Definicja w linii 18 pliku Terrain.cpp.

4.10.3.3 reset()

```
void Terrain::reset ( )
```

Resetuje mape.

Definicja w linii 37 pliku Terrain.cpp.

4.10.4 Dokumentacja atrybutów składowych

4.10.4.1 map

```
sf::Image Terrain::map
```

Obiekt mapy.

Definicja w linii 30 pliku Terrain.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

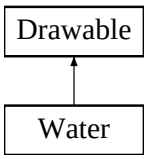
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Terrain.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Terrain.cpp

4.11 Dokumentacja klasy Water

Klasa wody.

`#include <Water.h>`

Diagram dziedziczenia dla Water



Metody publiczne

- Water ()
  Konstruktor.
- void update ()
  Aktualizuje obiekt wody co klatke.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override
  Rysuje obiekt na ekranie (dzidziczona z SFML)

4.11.1 Opis szczegółowy

Klasa wody.

Definicja w linii 7 pliku Water.h.

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 Water()

```
Water::Water ( )
```

Konstruktor.

Definicja w linii 2 pliku Water.cpp.

4.11.3 Dokumentacja funkcji składowych

4.11.3.1 draw()

```
void Water::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 64 pliku Water.cpp.

4.11.3.2 update()

```
void Water::update ( )
```

Aktualizuje obiekt wody co klatke.

Definicja w linii 35 pliku Water.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

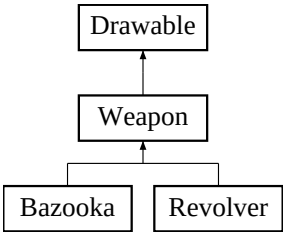
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Water.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Water.cpp

4.12 Dokumentacja klasy Weapon

Klasa bazowa broni.

```
#include <Weapon.h>
```

Diagram dziedziczenia dla Weapon



Metody publiczne

- Weapon ()=delete

  Konstruktor domyślny broni który jest usunięty.
- Weapon (float x, float y)

  Konstruktor broni.
- virtual ~Weapon ()=default

  Domyślny destruktor wirtualny broni.
- void setScale (float scale)

  Metoda ustawiająca skale broni.
- void setRotation (float rotation)

  Metoda ustawiająca rotacje broni.
- float getScale () const

  Metoda zwracająca skale broni.
- float getRotation () const

  Metoda zwracająca rotacje broni.
- sf::Sprite getSprite () const

  Metoda zwracająca obiekt sprite'a broni.
- float getPosX () const

  Zwraca współrzędną X broni.
- float getPosY () const

  Zwraca współrzędną Y broni.
- void setPosX (float x)

  Ustawia współrzędną X broni.
- void setPosY (float y)

  Ustawia współrzedną Y broni.
- void setBullet (Bullet *bullet)

  Ustawia pocisk broni na wskazany w parametrze obiekt.
- void setScaleVector (sf::Vector2f scale)

  Ustawia wektor skali broni.
- sf::Vector2f getScaleVector () const

  Zwraca wektor skali broni.
- Bullet * getBullet ()

  Zwraca wskaźnik na obiekt pocisku danej broni.
- void update ()

  Metoda aktualizująca obiekt broni co klatke.
- virtual void shoot (float angle)

  Wirtualna metoda sprawiająca, że broń wystrzeli.
- bool getIsShooting () const

  Zwraca wartość informującą o tym czy broń aktualnie strzela.
- void setIsShooting (bool isShooting)

  Ustawia stan strzelania broni na podany w parametrze.
- virtual void playShootSound ()

  Odtwarza dźwięk wystrzału broni.
- void setDamage (float damage)

  Ustawia obrażenia od danej broni.
- float getDamage () const

  Zwraca obrażenia danej broni.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Rysuje obiekt na ekranie (dzidziczona z SFML)

Atrybuty chronione

- sf::Sprite sprite

  Obiekt rysowany na ekranie.
- sf::Texture texture

  Tekstura obiektu.
- std::string texturePath

  Ścieżka do tekstury obiektu.
- Bullet ∗ bullet = nullptr
- float rotation = 0.0f

  Wartość rotacji broni.
- float scale = 0.2f

  Wartość skali broni.
- float posX

  Pozycja X broni na ekranie.
- float posY

  Pozycja Y broni na ekranie.
- sf::Vector2f scaleVector = { scale,scale }

  Wektor skali broni.
- bool isShooting = false

  Przechowuje informacje o tym czy broń strzela.
- float damage = 0.0f

  Obrażenia broni.

4.12.1 Opis szczegółowy

Klasa bazowa broni.

Definicja w linii 12 pliku Weapon.h.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 Weapon() [1/2]

```
Weapon::Weapon ( )  [delete]
```

Konstruktor domyślny broni który jest usunięty.

4.12.2.2 Weapon() [2/2]

```
Weapon::Weapon (
            float x,
            float y )
```

Konstruktor broni.

Parametry

x	- współrzędna x wyświetlenia miejsca broni
y	- współrzędna y wyświetlenia miejsca broni

Definicja w linii 4 pliku Weapon.cpp.

4.12.2.3 ~Weapon()

```
virtual Weapon::~Weapon ( )  [virtual], [default]
```

Domyślny destruktor wirtualny broni.

4.12.3 Dokumentacja funkcji składowych

4.12.3.1 draw()

```
void Weapon::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 131 pliku Weapon.cpp.

4.12.3.2 getBullet()

```
Bullet * Weapon::getBullet ( )
```

Zwraca wskaźnik na obiekt pocisku danej broni.

Zwraca

Wskaźnik obiekt pocisku danej broni

Definicja w linii 75 pliku Weapon.cpp.

4.12.3.3 getDamage()

```
float Weapon::getDamage ( ) const
```

Zwraca obrażenia danej broni.

Zwraca

obrażenia broni

Definicja w linii 126 pliku Weapon.cpp.

4.12.3.4 getIsShooting()

```
bool Weapon::getIsShooting ( ) const
```

Zwraca wartość informującą o tym czy broń aktualnie strzela.

Zwraca

wartość informującą o tym czy broń aktualnie strzela

Definicja w linii 107 pliku Weapon.cpp.

4.12.3.5 getPosX()

```
float Weapon::getPosX ( ) const
```

Zwraca współrzędną X broni.

Zwraca

współrzędną X broni

Definicja w linii 40 pliku Weapon.cpp.

4.12.3.6 getPosY()

```
float Weapon::getPosY ( ) const
```

Zwraca współrzędną Y broni.

Zwraca

współrzędną Y broni

Definicja w linii 45 pliku Weapon.cpp.

4.12.3.7 getRotation()

```
float Weapon::getRotation ( ) const
```

Metoda zwracająca rotacje broni.

Zwraca

Zwraca rotacje broni jako float

Definicja w linii 30 pliku Weapon.cpp.

4.12.3.8 getScale()

```
float Weapon::getScale ( ) const
```

Metoda zwracająca skale broni.

Zwraca

Zwraca skale broni

Definicja w linii 25 pliku Weapon.cpp.

4.12.3.9 getScaleVector()

```
sf::Vector2f Weapon::getScaleVector ( ) const
```

Zwraca wektor skali broni.

Zwraca

Wektor skali broni

Definicja w linii 70 pliku Weapon.cpp.

4.12.3.10 getSprite()

```
sf::Sprite Weapon::getSprite ( ) const
```

Metoda zwracająca obiekt sprite'a broni.

Zwraca

obiekt sf::Sprite broni

Definicja w linii 35 pliku Weapon.cpp.

4.12.3.11 playShootSound()

```
void Weapon::playShootSound ( )  [virtual]
```

Odtwarza dźwięk wystrzału broni.

Reimplementowana w Bazooka i Revolver.

Definicja w linii 117 pliku Weapon.cpp.

4.12.3.12 setBullet()

```
void Weapon::setBullet (
            Bullet * bullet )
```

Ustawia pocisk broni na wskazany w parametrze obiekt.

Parametry

bullet	- wskaźnik na obiekt pocisku

Definicja w linii 60 pliku Weapon.cpp.

4.12.3.13 setDamage()

```
void Weapon::setDamage (
            float damage )
```

Ustawia obrażenia od danej broni.

Parametry

damage	- obrażenia broni

Definicja w linii 121 pliku Weapon.cpp.

4.12.3.14 setIsShooting()

```
void Weapon::setIsShooting (
            bool isShooting )
```

Ustawia stan strzelania broni na podany w parametrze.

Parametry

isShooting	- czy broń strzela (true - tak, false - nie)

Definicja w linii 112 pliku Weapon.cpp.

4.12.3.15 setPosX()

```
void Weapon::setPosX (
            float x )
```

Ustawia współrzędną X broni.

Parametry

x	- Współrzędna x dla broni

Definicja w linii 50 pliku Weapon.cpp.

4.12.3.16 setPosY()

```
void Weapon::setPosY (
            float y )
```

Ustawia współrzędną Y broni.

Parametry

y	- współrzedna y dla broni

Definicja w linii 55 pliku Weapon.cpp.

4.12.3.17 setRotation()

```
void Weapon::setRotation (
            float rotation )
```

Metoda ustawiająca rotacje broni.

Parametry

rotation	- rotacja broni

Definicja w linii 20 pliku Weapon.cpp.

4.12.3.18 setScale()

```
void Weapon::setScale (
            float scale )
```

Metoda ustawiająca skale broni.

Parametry

scale	- skala broni

Definicja w linii 15 pliku Weapon.cpp.

4.12.3.19 setScaleVector()

```
void Weapon::setScaleVector (
            sf::Vector2f scale )
```

Ustawia wektor skali broni.

Parametry

scale	- wektor skali

Definicja w linii 65 pliku Weapon.cpp.

4.12.3.20 shoot()

```
void Weapon::shoot (
            float angle )  [virtual]
```

Wirtualna metoda sprawiająca, że broń wystrzeli.

Parametry

angle	- kąt pod jakim wystrzelony zostanie pocisk

Definicja w linii 90 pliku Weapon.cpp.

4.12.3.21 update()

```
void Weapon::update ( )
```

Metoda aktualizująca obiekt broni co klatke.

Definicja w linii 80 pliku Weapon.cpp.

4.12.4 Dokumentacja atrybutów składowych

4.12.4.1 bullet

```
Bullet* Weapon::bullet = nullptr  [protected]
```

/brief Wskaźnik na pocisk

Definicja w linii 177 pliku Weapon.h.

4.12.4.2 damage

```
float Weapon::damage = 0.0f  [protected]
```

Obrażenia broni.

Definicja w linii 212 pliku Weapon.h.

4.12.4.3 isShooting

```
bool Weapon::isShooting = false  [protected]
```

Przechowuje informacje o tym czy broń strzela.

Definicja w linii 207 pliku Weapon.h.

4.12.4.4 posX

```
float Weapon::posX  [protected]
```

Pozycja X broni na ekranie.

Definicja w linii 192 pliku Weapon.h.

4.12.4.5 posY

```
float Weapon::posY  [protected]
```

Pozycja Y broni na ekranie.

Definicja w linii 197 pliku Weapon.h.

4.12.4.6 rotation

```
float Weapon::rotation = 0.0f  [protected]
```

Wartość rotacji broni.

Definicja w linii 182 pliku Weapon.h.

4.12.4.7 scale

```
float Weapon::scale = 0.2f  [protected]
```

Wartość skali broni.

Definicja w linii 187 pliku Weapon.h.

4.12.4.8 scaleVector

```
sf::Vector2f Weapon::scaleVector = { scale,scale }  [protected]
```

Wektor skali broni.

Definicja w linii 202 pliku Weapon.h.

4.12.4.9 sprite

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

Obiekt rysowany na ekranie.

Definicja w linii 162 pliku Weapon.h.

4.12.4.10 texture

```
sf::Texture Weapon::texture  [protected]
```

Tekstura obiektu.

Definicja w linii 167 pliku Weapon.h.

4.12.4.11 texturePath

```
std::string Weapon::texturePath  [protected]
```

Ścieżka do tekstury obiektu.

Definicja w linii 172 pliku Weapon.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

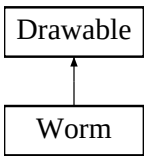
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Weapon.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Weapon.cpp

4.13 Dokumentacja klasy Worm

Klasa worma (gracza)

```
#include <Worm.h>
```

Diagram dziedziczenia dla Worm



Metody publiczne

- Worm ()

  Konstruktor domyślny.
- virtual ~Worm ()

  Destruktor.
- void update ()

  Aktualizuje obiekt worma co klatke.
- float left () const

  Zwraca lewą pozycje obiektu worma.
- float right () const

  Zwraca prawą pozycje obiektu worma.
- float top () const

  Zwraca górną pozycje obiektu worma.
- float bottom () const

  Zwraca dolna pozycje obiektu worma.
- float getWormX () const

  Zwraca współrzędną X pozycji worma na ekranie.
- float getWormY () const

  Zwraca współrzędną Y pozycji worma na ekranie.
- void stopMove ()
- void moveLeft ()

  Porusza worma w lewo.
- void moveRight ()

  Porusza worma w prawo.
- void jump ()

  Sprawia, że worm skacze.
- void damage (float dmg)

  Zadaje obrażenia podane w parametrze wormowi.
- bool isAlive ()

  Sprawdza czy worm jeszcze żyje.
- bool checkCollision (sf::Vector2f point)

  Sprawdza kolizje worma z danym punktem.
- int getOffsetY (sf::Vector2f point)

  Zwraca offset w płaszczyźnie Y względem punktu.
- float getScale () const

  Zwraca skale worma.
- sf::Sprite getSprite () const

  Zwraca obiekt rysowanego worma.
- void setWeapon (Weapon *weapon)

  Ustawia wormowi daną broń na tą do której podamy wskaźnik.
- Weapon * getWeapon () const

  Zwraca wskaźnik na obiekt aktualnie trzymanej broni przez worma.
- void deleteWeapon ()

  Usuwa broń którą trzyma worm.
- void setColMap (sf::Image *image)

  Ustawia kolizje mapy.
- sf::Text getDebugTxt ()

  Zwraca tekst dla debbuge'a.
- bool isLookingOnLeft () const

  Sprawdza czy worm patrzy w lewo.

- bool hasWeapon () const

  Sprawdza czy worm ma broń
- void setTeam (team t)

  Ustawia team dla danego worma.
- void lookLeft ()

  Sprawia, że worm patrzy w lewo.
- void lookRight ()

  Sprawia, że worm patrzy w prawo.
- void setChoosen ()

  Ustawia worma jako wybranego aktualnie do sterowania.
- void setNormal ()

  Ustawia worma jako normalnego (nie sterujemy nim)

Atrybuty publiczne

- sf::Vector2f collisionPoints [8]

  Tablica wektorów punktów kolizji worma.

Metody chronione

- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Rysuje obiekt na ekranie (dzidziczona z SFML)

4.13.1 Opis szczegółowy

Klasa worma (gracza)

Definicja w linii 20 pliku Worm.h.

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 Worm()

```
Worm::Worm ( )
```

Konstruktor domyślny.

Definicja w linii 8 pliku Worm.cpp.

4.13.2.2 ∼Worm()

```
Worm::∼Worm ( )  [virtual]
```

Destruktor.

Definicja w linii 65 pliku Worm.cpp.

4.13.3 Dokumentacja funkcji składowych

4.13.3.1 bottom()

```
float Worm::bottom ( ) const
```

Zwraca dolna pozycje obiektu worma.

Zwraca

dolna pozycje obiektu worma

Definicja w linii 239 pliku Worm.cpp.

4.13.3.2 checkCollision()

```
bool Worm::checkCollision (
            sf::Vector2f point )
```

Sprawdza kolizje worma z danym punktem.

Parametry

point	- punkt do sprawdznia kolizji

Zwraca

true - jeśli kolizja nastąpi lub false - jeśli nie nastąpi

Definicja w linii 197 pliku Worm.cpp.

4.13.3.3 damage()

```
void Worm::damage (
            float dmg )
```

Zadaje obrażenia podane w parametrze wormowi.

Parametry

dmg	- obrażenia dla worma

Definicja w linii 351 pliku Worm.cpp.

4.13.3.4 deleteWeapon()

```
void Worm::deleteWeapon ( )
```

Usuwa broń którą trzyma worm.

Definicja w linii 191 pliku Worm.cpp.

4.13.3.5 draw()

```
void Worm::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [protected]
```

Rysuje obiekt na ekranie (dzidziczona z SFML)

Parametry

target	
states	

Definicja w linii 211 pliku Worm.cpp.

4.13.3.6 getDebugTxt()

```
sf::Text Worm::getDebugTxt ( )
```

Zwraca tekst dla debbuge'a.

Zwraca

tekst dla debbuge'a

Definicja w linii 361 pliku Worm.cpp.

4.13.3.7 getOffsetY()

```
int Worm::getOffsetY (
            sf::Vector2f point )
```

Zwraca offset w płaszczyźnie Y względem punktu.

Parametry

point	

Zwraca

offset w płaszczyźnie Y

Definicja w linii 153 pliku Worm.cpp.

4.13.3.8 getScale()

```
float Worm::getScale ( ) const
```

Zwraca skale worma.

Zwraca

skale worma

Definicja w linii 168 pliku Worm.cpp.

4.13.3.9 getSprite()

```
sf::Sprite Worm::getSprite ( ) const
```

Zwraca obiekt rysowanego worma.

Zwraca

Definicja w linii 173 pliku Worm.cpp.

4.13.3.10 getWeapon()

```
Weapon * Worm::getWeapon ( ) const
```

Zwraca wskaźnik na obiekt aktualnie trzymanej broni przez worma.

Zwraca

wskaźnik na obiekt broni lub nullptr gdy worm nie trzyma żadnej broni

Definicja w linii 186 pliku Worm.cpp.

4.13.3.11 getWormX()

```
float Worm::getWormX ( ) const
```

Zwraca współrzędną X pozycji worma na ekranie.

Zwraca

współrzędną X pozycji worma na ekranie

Definicja w linii 244 pliku Worm.cpp.

4.13.3.12 getWormY()

```
float Worm::getWormY ( ) const
```

Zwraca współrzędną Y pozycji worma na ekranie.

Zwraca

współrzędną Y pozycji worma na ekranie

Definicja w linii 249 pliku Worm.cpp.

4.13.3.13 hasWeapon()

```
bool Worm::hasWeapon ( ) const
```

Sprawdza czy worm ma broń

Zwraca

true - jeśli worm ma broń lub false - jeśli jej nie ma

Definicja w linii 381 pliku Worm.cpp.

4.13.3.14 isAlive()

```
bool Worm::isAlive ( )
```

Sprawdza czy worm jeszcze żyje.

Zwraca

true - jeśli worm żyje lub false - jeśli nie żyje

Definicja w linii 356 pliku Worm.cpp.

4.13.3.15 isLookingOnLeft()

```
bool Worm::isLookingOnLeft ( ) const
```

Sprawdza czy worm patrzy w lewo.

Zwraca

true - jeśli parzy w lewo lub false - jeśli parzy w prawo

Definicja w linii 376 pliku Worm.cpp.

4.13.3.16 jump()

```
void Worm::jump ( )
```

Sprawia, że worm skacze.

Definicja w linii 340 pliku Worm.cpp.

4.13.3.17 left()

```
float Worm::left ( ) const
```

Zwraca lewą pozycje obiektu worma.

Zwraca

lewą pozycje obiektu worma

Definicja w linii 227 pliku Worm.cpp.

4.13.3.18 lookLeft()

```
void Worm::lookLeft ( )
```

Sprawia, że worm patrzy w lewo.

Definicja w linii 266 pliku Worm.cpp.

4.13.3.19 lookRight()

```
void Worm::lookRight ( )
```

Sprawia, że worm patrzy w prawo.

Definicja w linii 279 pliku Worm.cpp.

4.13.3.20 moveLeft()

```
void Worm::moveLeft ( )
```

Porusza worma w lewo.

Definicja w linii 304 pliku Worm.cpp.

4.13.3.21 moveRight()

```
void Worm::moveRight ( )
```

Porusza worma w prawo.

Definicja w linii 322 pliku Worm.cpp.

4.13.3.22 right()

```
float Worm::right ( ) const
```

Zwraca prawą pozycje obiektu worma.

Zwraca

prawą pozycje obiektu worma

Definicja w linii 231 pliku Worm.cpp.

4.13.3.23 setChoosen()

```
void Worm::setChoosen ( )
```

Ustawia worma jako wybranego aktualnie do sterowania.

Definicja w linii 293 pliku Worm.cpp.

4.13.3.24 setColMap()

```
void Worm::setColMap (
            sf::Image * image )
```

Ustawia kolizje mapy.

Parametry

image	- obraz mapy

Definicja w linii 149 pliku Worm.cpp.

4.13.3.25 setNormal()

```
void Worm::setNormal ( )
```

Ustawia worma jako normalnego (nie sterujemy nim)

Definicja w linii 299 pliku Worm.cpp.

4.13.3.26 setTeam()

```
void Worm::setTeam (
            team t )
```

Ustawia team dla danego worma.

Parametry

t	- team dla worma

Definicja w linii 386 pliku Worm.cpp.

4.13.3.27 setWeapon()

```
void Worm::setWeapon (
            Weapon * weapon )
```

Ustawia wormowi daną broń na tą do której podamy wskaźnik.

Parametry

weapon	- wskaźnik na broń dla worma

Definicja w linii 178 pliku Worm.cpp.

4.13.3.28 stopMove()

```
void Worm::stopMove ( )
```

Zatrzymuje worma

Definicja w linii 254 pliku Worm.cpp.

4.13.3.29 top()

```
float Worm::top ( ) const
```

Zwraca górną pozycje obiektu worma.

Zwraca

górną pozycje obiektu worma

Definicja w linii 235 pliku Worm.cpp.

4.13.3.30 update()

```
void Worm::update ( )
```

Aktualizuje obiekt worma co klatke.

Definicja w linii 70 pliku Worm.cpp.

4.13.4 Dokumentacja atrybutów składowych

4.13.4.1 collisionPoints

```
sf::Vector2f Worm::collisionPoints[8]
```

Tablica wektorów punktów kolizji worma.

Definicja w linii 136 pliku Worm.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Worm.h
- D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Worm.cpp

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bazooka.cpp

```
#include "Bazooka.h"
#include "GameWindow.h"
```

5.2 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bazooka.h

```
#include "Weapon.h"
```

Komponenty

- class Bazooka

  Klasa bazooki.

5.3 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bullet.cpp

```
#include "Bullet.h"
```

5.4 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Bullet.h

```
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Sprite.hpp>
#include <SFML/Graphics/Texture.hpp>
#include <SFML/Graphics/RenderTarget.hpp>
```

Komponenty

- class Bullet

  Klasa pocisku.

5.5 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Button.cpp

```
#include "Button.h"
```

5.6 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Button.h

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

Komponenty

- class Button

  Klasa przycisku.

5.7 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/FPSCounter.cpp

```
#include "FPSCounter.h"
#include <iostream>
#include "GameWindow.h"
```

5.8 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/FPSCounter.h

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

Komponenty

- class FPSCounter

  Klasa licznika FPS'ów.

5.9 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameEvent.cpp

```
#include "GameEvent.h"
#include <iostream>
#include "Revolver.h"
```

5.10 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameEvent.h

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

Komponenty

- class GameEvent

  Klasa zdarzeń gry.

5.11 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameSound.cpp

```
#include "GameSound.h"
```

5.12 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameSound.h

```
#include <string>
#include <SFML/Audio.hpp>
#include <iostream>
```

Komponenty

- class GameSound

  Klasa dźwięków gry.

5.13 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameWindow.cpp

```
#include "GameWindow.h"
#include "Bullet.h"
```

5.14 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/GameWindow.h

```
#include <string>
#include <SFML/Graphics.hpp>
#include <vector>
#include "FPSCounter.h"
#include "Worm.h"
#include "GameSound.h"
#include "Terrain.h"
#include "Bullet.h"
#include "Weapon.h"
#include "Bazooka.h"
#include "Water.h"
#include "Menu.h"
```

Komponenty

- class GameWindow

  Klasa okna głównego.

5.15 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Menu.cpp

```
#include "Menu.h"
#include <iostream>
#include "GameSound.h"
#include "GameWindow.h"
```

5.16 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Menu.h

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

Komponenty

- class Menu

  Klasa ggo menu.

Wyliczenia

- enum game_states {
  GAME, PAUSE, MENU, HELP,
  HELP_PAUSE, RED_WIN, BLUE_WIN, EXIT,
  KEYBOARD, GAMEPAD }

  Typ wyliczeniowy dla stany.

5.16.1 Dokumentacja typów wyliczanych

5.16.1.1 game_states

```
enum game_states
```

Typ wyliczeniowy dla stany.

Wartości wyliczeń

GAME	
PAUSE	
MENU	
HELP	
HELP_PAUSE	
RED_WIN	
BLUE_WIN	
EXIT	
KEYBOARD	
GAMEPAD	

Definicja w linii 9 pliku Menu.h.

5.17 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Revolver.cpp

```
#include "Revolver.h"
#include "GameSound.h"
#include "GameWindow.h"
```

5.18 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Revolver.h

```
#include "Weapon.h"
```

Komponenty

- class Revolver

  Klasa rewolwera.

5.19 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Source.cpp

```
#include "GameEvent.h"
#include "GameWindow.h"
#include "GameSound.h"
```

Funkcje

- int main ()

5.19.1 Dokumentacja funkcji

5.19.1.1 main()

```
int main ( )
```

Definicja w linii 6 pliku Source.cpp.

5.20 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Terrain.cpp

```
#include "Terrain.h"
#include <iostream>
```

5.21 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Terrain.h

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

Komponenty

- class Terrain

  Klasa terenu(mapy)

5.22 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Water.cpp

```
#include "Water.h"
```

5.23 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Water.h

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

Komponenty

- class Water

  Klasa wody.

5.24 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Weapon.cpp

```
#include "Weapon.h"
#include <iostream>
```

5.25 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Weapon.h

```
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Sprite.hpp>
#include <SFML/Graphics/Texture.hpp>
#include "Bullet.h"
```

Komponenty

- class Weapon

  Klasa bazowa broni.

5.26 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Worm.cpp

```
#include "Worm.h"
#include <string>
#include <iostream>
#include "GameWindow.h"
#include "GameEvent.h"
```

5.27 Dokumentacja pliku D:/Projekty/C++/Programowanie w C2/Worms 2D/Worms 2D/Worm.h

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

Komponenty

- class Worm

  Klasa worma (gracza)

Wyliczenia

- enum collision {
  UP, UP_LEFT, UP_RIGHT, DOWN,
  DOWN_LEFT, DOWN_RIGHT, LEFT, RIGHT }

  Typ wyliczeniowy dla kolizji.
- enum team { RED, BLUE }

  Typ wyliczeniowy dla drużyny.

5.27.1 Dokumentacja typów wyliczanych

5.27.1.1 collision

```
enum collision
```

Typ wyliczeniowy dla kolizji.

Wartości wyliczeń

UP	
UP_LEFT	
UP_RIGHT	
DOWN	
DOWN_LEFT	
DOWN_RIGHT	
LEFT	
RIGHT	

Definicja w linii 10 pliku Worm.h.

5.27.1.2 team

```
enum team
```

Typ wyliczeniowy dla drużyny.

Wartości wyliczeń

RED	
BLUE	

Definicja w linii 15 pliku Worm.h.

Skorowidz