# Boomberman

Wygenerowano przez Doxygen 1.8.14

# Spis treści

# Rozdział 1

# Indeks struktur danych

## 1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

# Rozdział 2

# Indeks plików

## 2.1 Lista plików

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

# Rozdział 3

# Dokumentacja struktur danych

## 3.1 Dokumentacja struktury Bomb

`#include <enumsStructsMacros.h>`

### Pola danych

- int size
- int fireCross [4]
- XY coords
- double explosionTimer
- double kickTimer
- Direction kickDirection
- bombState statFlag

### 3.1.1 Opis szczegółowy

Bomba i jej dane

### 3.1.2 Dokumentacja pól

#### 3.1.2.1 coords

`XY Bomb::coords`

Koordynaty bomby wzgledem mapy.

#### 3.1.2.2 explosionTimer

```
double Bomb::explosionTimer
```

Czas do eksplozji lub czas po eksplozji.

#### 3.1.2.3 fireCross

```
int Bomb::fireCross[4]
```

Gdzie po wycuchu siegnal ogien bomby w 4 kierunkach.

#### 3.1.2.4 kickDirection

```
Direction Bomb::kickDirection
```

Kierunek przesuwania bomby.

#### 3.1.2.5 kickTimer

```
double Bomb::kickTimer
```

Kiedy ostatni raz bomba byla przesunieta.

#### 3.1.2.6 size

```
int Bomb::size
```

Zasieg bomby.

#### 3.1.2.7 statFlag

```
bombState Bomb::statFlag
```

Status bomby.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- enumsStructsMacros.h

## 3.2 Dokumentacja struktury Enemy

```
#include <enumsStructsMacros.h>
```

### Pola danych

- Status status
- XY coords
- Direction direction
- EnemyType type
- float rottingTime

## 3.2.1 Opis szczegółowy

Przeciwnik i jego dane

## 3.2.2 Dokumentacja pól

### 3.2.2.1 coords

```
XY Enemy::coords
```

Koordynaty przeciwnika wzgledem mapy.

### 3.2.2.2 direction

```
Direction Enemy::direction
```

Kierunek poruszania sie przeciwnika.

### 3.2.2.3 rottingTime

```
float Enemy::rottingTime
```

Jak dlugo nalezy rysowac zwloki przeciwnika po jego smierci.

### 3.2.2.4 status

```
Status Enemy::status
```

Status przeciwnika.

### 3.2.2.5 type

```
EnemyType Enemy::type
```

Typ przeciwnika.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- enumsStructsMacros.h

## 3.3 Dokumentacja struktury Player

`#include <enumsStructsMacros.h>`

### Pola danych

- XY coords
- Status status
- int upgrades [5]
- int score
- int timeLeft
- double brokenShieldTimer
- Bomb bombList [MAX_BOMBS]
- int newestBomb
- int oldestBomb

### 3.3.1 Opis szczegółowy

Gracz i jego dane

### 3.3.2 Dokumentacja pól

#### 3.3.2.1 bombList

`Bomb Player::bombList[MAX_BOMBS]`

Tablica zawierajaca bomby postawione przez gracza.

#### 3.3.2.2 brokenShieldTimer

`double Player::brokenShieldTimer`

Czas po zniszczeniu 'tarczy' przez jaki bedzie ona jeszcze aktywna .

#### 3.3.2.3 coords

`XY Player::coords`

Koordynaty gracza wzgledem mapy.

#### 3.3.2.4 newestBomb

```
int Player::newestBomb
```

Indeks najpozniej polozonej i jeszcze aktywnej bomby w tablicy bomb.

#### 3.3.2.5 oldestBomb

```
int Player::oldestBomb
```

Indeks najwczesniej polozonej i jeszcze aktywnej bomby w tablicy bomb.

#### 3.3.2.6 score

```
int Player::score
```

Punkty gracza.

#### 3.3.2.7 status

```
Status Player::status
```

Status gracza.

#### 3.3.2.8 timeLeft

```
int Player::timeLeft
```

Czas, po ktorym gracz zacznie tracic punkty, az sie wyzeruja, po czym gracz umrze. Ten czas jest resetowany po przegraniu lub wygraniu planszy.

#### 3.3.2.9 upgrades

```
int Player::upgrades[5]
```

Ulepszenia gracza.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- enumsStructsMacros.h

## 3.4 Dokumentacja struktury tile

### Pola danych

- tileType type
- int fireLayers
- upgrades upgrade

### 3.4.1 Dokumentacja pól

#### 3.4.1.1 fireLayers

```
int tile::fireLayers
```

Ile jest warstw ognia na kafelku.

#### 3.4.1.2 type

```
tileType tile::type
```

Typ kafelka.

#### 3.4.1.3 upgrade

```
upgrades tile::upgrade
```

Jakie ulepszenie jest na kafelku.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- enumsStructsMacros.h

## 3.5 Dokumentacja struktury XY

### Pola danych

- int x
- int y

### 3.5.1 Dokumentacja pól

#### 3.5.1.1 x

```
int XY::x
```

Koordynat x.

#### 3.5.1.2 y

```
int XY::y
```

Koordynat y.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- enumsStructsMacros.h

# Rozdział 4

# Dokumentacja plików

## 4.1 Dokumentacja pliku draw.c

```
#include "enumsStructsMacros.h"
#include "draw.h"
#include <GLFW\glfw3.h>
#include <stdlib.h>
#include "mapMemAndGen.h"
```

### Funkcje

- void initGrid (tile **bitmap, int gridSize, int howManyBoxes, int howManyUpgrades)
- void resetDrawStatics ()
- void setGlLists (int listArray[6], int gridSize)
- void unSetGlList (int listArray[6])
- void drawGridBase (int gridSize)
- void drawGrid (tile **bitmap, int gridSize)
- void drawBombs (Player *player, int gridSize)
- void drawBomb (double x, double y, double sizeX, double sizeY, Bomb *bomb)
- void drawUpgrade (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawSquare (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawPortal (double x, double y, double sizeX, double sizeY, int reset)
- void drawPlayer (Player *player, int gridSize)
- void drawScoreboard ()
- void drawEnemyBlind (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemyRandom (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemychargerCalm (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemychargerUpset (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemy (double x, double y, double sizeX, double sizeY, Enemy *enemy)
- void drawAllEnemies (Enemy enemyArray[MAX_ENEMIES_PER_MAP], int gridSize, int howMany)
- void drawPauseScreen ()

### 4.1.1 Dokumentacja funkcji

#### 4.1.1.1 drawAllEnemies()

```
void drawAllEnemies (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            int gridSize,
            int howMany )
```

Rysuje panel obok mapy, na ktorym wypisuje punkty gracza, najwyzsze dotychczasowe punkty oraz pozostaly czas gry

**Parametry**

| | |
|---|---|
| *enemyArray* | tablica przechowujaca dane wszystkich przeciwnikow |
| *gridSize* | wielkosc mapy (NxN) |
| *howMany* | ile przeciwnikow jest do narysowania |

#### 4.1.1.2 drawBomb()

```
void drawBomb (
            double x,
            double y,
            double sizeX,
            double sizeY,
            Bomb * bomb )
```

Rysuje pojedyncza bombe o okreslonym okreslonym rozmiarze w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |
| *bomb* | wskaznik na bombe do narysowania |

#### 4.1.1.3 drawBombs()

```
void drawBombs (
            Player * player,
            int gridSize )
```

Wywoluje rysowanie wszystkich bomb w okreslonych miejscach

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza (gracz przechowuje bomby) |
| *gridSize* | wielkosc mapy (NxN) |

#### 4.1.1.4 drawEnemy()

```
void drawEnemy (
            double x,
            double y,
            double sizeX,
            double sizeY,
            Enemy * enemy )
```

Wywoluje funkcje rysowania przeciwnika wzgledem jego typu w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *x* | Koordynat szerokosci wzgledem okna. |
| *y* | Kordynat wysokosci wzgledem okna. |
| *sizeX* | Szerokosc obiektu wzgledem okna. |
| *sizeY* | Wysokosc obiektu wzgledem okna. |
| *enemy* | Wskaznik na przeciwnika do narysowania. |

#### 4.1.1.5 drawEnemyBlind()

```
void drawEnemyBlind (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'blind' o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.6 drawEnemychargerCalm()

```
void drawEnemychargerCalm (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'charger' w pierwszej jego fazie o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.7 drawEnemychargerUpset()

```
void drawEnemychargerUpset (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'charger' w jego drugiej fazie o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.8 drawEnemyRandom()

```
void drawEnemyRandom (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'random' o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.9 drawGrid()

```
void drawGrid (
            tile ** bitmap,
            int gridSize )
```

Funkcja rysujaca zmienna czesc mapy - pudelka, ulepszenia, ogien, zniszczone pudelka, oraz portal

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy (NxN) |

#### 4.1.1.10 drawGridBase()

```
void drawGridBase (
            int gridSize )
```

Funkcja rysujaca podstawe mapy - tlo, sciezki, kolumny

**Parametry**

| | |
|---|---|
| *gridSize* | wielkosc mapy (NxN) |

#### 4.1.1.11 drawPauseScreen()

```
void drawPauseScreen ( )
```

Rysuje tlo ekran pauzy

#### 4.1.1.12 drawPlayer()

```
void drawPlayer (
            Player * player,
            int gridSize )
```

Rysuje gracza w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy (NxN) |

#### 4.1.1.13 drawPortal()

```
void drawPortal (
            double x,
            double y,
            double sizeX,
            double sizeY,
            int reset )
```

Rysuje portal o okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.14 drawScoreboard()

```
void drawScoreboard ( )
```

Rysuje tlo panel obok mapy, na ktorym wypisane beda punkty gracza, najwyzsze dotychczasowe punkty oraz pozostaly czas gry

#### 4.1.1.15 drawSquare()

```
void drawSquare (
             float R,
             float G,
             float B,
             double x,
             double y,
             double sizeX,
             double sizeY )
```

Rysuje kwadrat o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.16 drawUpgrade()

```
void drawUpgrade (
             float R,
             float G,
             float B,
             double x,
             double y,
             double sizeX,
             double sizeY )
```

Rysuje ulepszenie o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.1.1.17 initGrid()

```
void initGrid (
            tile ** bitmap,
            int gridSize,
            int howManyBoxes,
            int howManyUpgrades )
```

Inicjalizuje rzeczy wymagajace inicjalizacji z tego pliku

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy (NxN) |
| *howManyBoxes* | ile na mapie znajduje sie pudel |
| *howManyUpgrades* | ile na mapie znajduje sie ulepszen |

#### 4.1.1.18 resetDrawStatics()

```
void resetDrawStatics ( )
```

Wywoluje funkcje zawierajace zmienne statyczne tak, by je zresetowac

#### 4.1.1.19 setGlLists()

```
void setGlLists (
            int listArray[6],
            int gridSize )
```

Ustawia glListy dla optymalniejszego rysowania

**Parametry**

| | |
|---|---|
| *listArray* | Tablica przechowujaca indeksy glList. |
| *gridSize* | wielkosc mapy |

#### 4.1.1.20 unSetGlList()

```
void unSetGlList (
            int listArray[6] )
```

Usuwa stworzone wczesniej glListy

**Parametry**

| | |
|---|---|
| *listArray* | Tablica przechowujaca indeksy glList. |

## 4.2 Dokumentacja pliku draw.h

### Funkcje

- void initGrid (tile **bitmap, int gridSize, int howManyBoxes, int howManyUpgrades)
- void resetDrawStatics ()
- void setGlLists (int listArray[6], int gridSize)
- void unSetGlList (int listArray[6])
- void drawGridBase (int gridSize)
- void drawGrid (tile **bitmap, int gridSize)
- void drawBombs (Player *player, int gridSize)
- void drawBomb (double x, double y, double sizeX, double sizeY, Bomb *bomb)
- void drawUpgrade (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawSquare (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawPortal (double x, double y, double sizeX, double sizeY, int reset)
- void drawPlayer (Player *player, int gridSize)
- void drawScoreboard ()
- void drawEnemyBlind (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemyRandom (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemychargerCalm (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemychargerUpset (float R, float G, float B, double x, double y, double sizeX, double sizeY)
- void drawEnemy (double x, double y, double sizeX, double sizeY, Enemy *enemy)
- void drawAllEnemies (Enemy enemyArray[MAX_ENEMIES_PER_MAP], int gridSize, int howMany)
- void drawPauseScreen ()

### 4.2.1 Dokumentacja funkcji

#### 4.2.1.1 drawAllEnemies()

```
void drawAllEnemies (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            int gridSize,
            int howMany )
```

Rysuje panel obok mapy, na ktorym wypisuje punkty gracza, najwyzsze dotychczasowe punkty oraz pozostaly czas gry

**Parametry**

| | |
|---|---|
| *enemyArray* | tablica przechowujaca dane wszystkich przeciwnikow |
| *gridSize* | wielkosc mapy (NxN) |
| *howMany* | ile przeciwnikow jest do narysowania |

#### 4.2.1.2 drawBomb()

```
void drawBomb (
            double x,
            double y,
            double sizeX,
            double sizeY,
            Bomb * bomb )
```

Rysuje pojedyncza bombe o okreslonym okreslonym rozmiarze w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |
| *bomb* | wskaznik na bombe do narysowania |

#### 4.2.1.3 drawBombs()

```
void drawBombs (
            Player * player,
            int gridSize )
```

Wywoluje rysowanie wszystkich bomb w okreslonych miejscach

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza (gracz przechowuje bomby) |
| *gridSize* | wielkosc mapy (NxN) |

#### 4.2.1.4 drawEnemy()

```
void drawEnemy (
            double x,
            double y,
            double sizeX,
            double sizeY,
            Enemy * enemy )
```

Wywoluje funkcje rysowania przeciwnika wzgledem jego typu w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *x* | Koordynat szerokosci wzgledem okna. |
| *y* | Kordynat wysokosci wzgledem okna. |
| *sizeX* | Szerokosc obiektu wzgledem okna. |
| *sizeY* | Wysokosc obiektu wzgledem okna. |
| *enemy* | Wskaznik na przeciwnika do narysowania. |

#### 4.2.1.5 drawEnemyBlind()

```
void drawEnemyBlind (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'blind' o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.6 drawEnemychargerCalm()

```
void drawEnemychargerCalm (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'charger' w pierwszej jego fazie o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.7 drawEnemychargerUpset()

```
void drawEnemychargerUpset (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'charger' w jego drugiej fazie o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.8 drawEnemyRandom()

```
void drawEnemyRandom (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje model przeciwnika typu 'random' o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.9 drawGrid()

```
void drawGrid (
            tile ** bitmap,
            int gridSize )
```

Funkcja rysujaca zmienna czesc mapy - pudelka, ulepszenia, ogien, zniszczone pudelka, oraz portal

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy (NxN) |

#### 4.2.1.10 drawGridBase()

```
void drawGridBase (
            int gridSize )
```

Funkcja rysujaca podstawe mapy - tlo, sciezki, kolumny

**Parametry**

| | |
|---|---|
| *gridSize* | wielkosc mapy (NxN) |

#### 4.2.1.11 drawPauseScreen()

```
void drawPauseScreen ( )
```

Rysuje tlo ekran pauzy

#### 4.2.1.12 drawPlayer()

```
void drawPlayer (
            Player * player,
            int gridSize )
```

Rysuje gracza w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy (NxN) |

#### 4.2.1.13 drawPortal()

```
void drawPortal (
            double x,
            double y,
            double sizeX,
            double sizeY,
            int reset )
```

Rysuje portal o okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.14 drawScoreboard()

```
void drawScoreboard ( )
```

Rysuje tlo panel obok mapy, na ktorym wypisane beda punkty gracza, najwyzsze dotychczasowe punkty oraz pozostaly czas gry

#### 4.2.1.15 drawSquare()

```
void drawSquare (
            float R,
            float G,
            float B,
```

```
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje kwadrat o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.16 drawUpgrade()

```
void drawUpgrade (
            float R,
            float G,
            float B,
            double x,
            double y,
            double sizeX,
            double sizeY )
```

Rysuje ulepszenie o okreslonym kolorze, okreslonym rozmiarze i w okreslonym miejscu

**Parametry**

| | |
|---|---|
| *R* | Skladowa koloru RGB |
| *G* | Skladowa koloru RGB |
| *B* | Skladowa koloru RGB |
| *x* | koordynat szerokosci wzgledem okna |
| *y* | koordynat wysokosci wzgledem okna |
| *sizeX* | szerokosc obiektu wzgledem okna |
| *sizeY* | wysokosc obiektu wzgledem okna |

#### 4.2.1.17 initGrid()

```
void initGrid (
            tile ** bitmap,
            int gridSize,
```

```
            int howManyBoxes,
            int howManyUpgrades )
```

Inicjalizuje rzeczy wymagajace inicjalizacji z tego pliku

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy (NxN) |
| *howManyBoxes* | ile na mapie znajduje sie pudel |
| *howManyUpgrades* | ile na mapie znajduje sie ulepszen |

#### 4.2.1.18 resetDrawStatics()

```
void resetDrawStatics ( )
```

Wywoluje funkcje zawierajace zmienne statyczne tak, by je zresetowac

#### 4.2.1.19 setGlLists()

```
void setGlLists (
            int listArray[6],
            int gridSize )
```

Ustawia glListy dla optymalniejszego rysowania

**Parametry**

| | |
|---|---|
| *listArray* | Tablica przechowujaca indeksy glList. |
| *gridSize* | wielkosc mapy |

#### 4.2.1.20 unSetGlList()

```
void unSetGlList (
            int listArray[6] )
```

Usuwa stworzone wczesniej glListy

**Parametry**

| | |
|---|---|
| *listArray* | Tablica przechowujaca indeksy glList. |

## 4.3 Dokumentacja pliku Enemies.c

```
#include "enumsStructsMacros.h"
#include "Enemies.h"
#include <stdlib.h>
#include <GLFW/glfw3.h>
```

### Funkcje

- void initEnemies (Enemy enemyArray[MAX_ENEMIES_PER_MAP], tile *bitmap, int howMany, int gridSize)
- void resetEnemyStatics ()
- void addEnemy (Enemy enemyArray[MAX_ENEMIES_PER_MAP], int ID, EnemyType type, XY coords, Direction dir)
- void moveBlind (Enemy *enemy, tile **bitmap, int gridSize)
- void moveRandom (Enemy *enemy, tile **bitmap, int gridSize)
- int pointTowardsPlayerIfPossible (Enemy *enemy, Player *player)
- void moveCharger (Enemy *enemy, Player *player, tile **bitmap, int gridSize)
- int moveEnemy (Enemy *enemy, int charge, tile **bitmap, int gridSize)
- int manageEnemiesAndScore (Enemy enemyArray[MAX_ENEMIES_PER_MAP], Player *player, tile **bitmap, int howMany, int gridSize, int reset)
- void dropEnemies (Enemy enemyArray[MAX_ENEMIES_PER_MAP], tile **bitmap, int howMany, int gridSize)

### 4.3.1 Dokumentacja funkcji

#### 4.3.1.1 addEnemy()

```
void addEnemy (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            int ID,
            EnemyType type,
            XY coords,
            Direction dir )
```

Dodaje nowego przeciwnika do tablicy i ustawia jego dane.

**Parametry**

| | |
|---|---|
| *enemyArray* | Tablica przechowujaca dane wszystkich przeciwnikow. |
| *ID* | Indeks nowego przeciwnika w tablicy przeciwnikow. |
| *type* | Typ przeciwnika. |
| *coords* | Koordynaty nowegp przeciwnika wzgledem mapy. |
| *dir* | Kiedunek poruszania sie nowego przeciwnika. |

#### 4.3.1.2 dropEnemies()

```
void dropEnemies (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            tile ** bitmap,
            int howMany,
            int gridSize )
```

Stawia nowych przeciwnikow na planszy.

**Parametry**

| | |
|---|---|
| *enemyArray* | Tablica przechowujaca dane wszystkich przeciwnikow. |
| *bitmap* | Wskaznik na mape gry. |
| *howMany* | Ile przeciwnikow jest do wygenerowania. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.3.1.3 initEnemies()

```
void initEnemies (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            tile * bitmap,
            int howMany,
            int gridSize )
```

Inicjalizuje rzeczy wymagajace inicjalizacji z tego pliku.

**Parametry**

| | |
|---|---|
| *enemyArray* | Tablica przechowujaca dane wszystkich przeciwnikow. |
| *bitmap* | Wskaznik na mape gry. |
| *howMany* | Ile przeciwnikow jest do wygenerowania. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.3.1.4 manageEnemiesAndScore()

```
int manageEnemiesAndScore (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            Player * player,
            tile ** bitmap,
            int howMany,
            int gridSize,
            int reset )
```

Obsluguje przeciwnikow, tj. przesuwa ich, sprawdza czy nie zostali zabici przez gracza (stoja na ogniu), dodaje punkty graczowi, jesli przeciwnik umiera i sprawdza, czy ktorys z przeciwnikow jeszcze zyje.

**Parametry**

| | |
|---|---|
| *enemyArray* | tablica przechowujaca dane wszystkich przeciwnikow. |
| *player* | Wskaznik na gracza. |
| *bitmap* | Wskaznik na mape gry. |
| *howMany* | Ile jest przeciwnikow. |
| *gridSize* | Wielkosc mapy (NxN). |
| *reset* | Jesli 1 to nalezy zresetowac wartosci statycze w funkcji i z niej wyjsc, bez wykonywania reszty jej ciala. |

**Zwraca**

Liczba 0 kiedy zyje przynajmniej jeden przeciwnik, liczba 1 jesli wszyscy sa martwi.

#### 4.3.1.5 moveBlind()

```
void moveBlind (
            Enemy * enemy,
            tile ** bitmap,
            int gridSize )
```

Przesuwa przeciwnika typu 'blind' lub przeciwnika typu 'charger' w pierwszej jego fazie.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.3.1.6 moveCharger()

```
void moveCharger (
            Enemy * enemy,
            Player * player,
            tile ** bitmap,
            int gridSize )
```

Przesuwa przeciwnika typu 'charger' w jego drugiej fazie.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *player* | Wskaznik na gracza. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.3.1.7 moveEnemy()

```
int moveEnemy (
            Enemy * enemy,
            int rush,
            tile ** bitmap,
            int gridSize )
```

Probuje przesunac przeciwnika w jego kierunku o jeden kafelek.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *rush* | Przelacznik, czy przeciwnik ma sie przebijac przez pudelka. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

**Zwraca**

Liczba 0 jesli sie nie udalo przesunac przeciwnika, liczba 1 w przypadku kiedy sie udalo, 2 jesli kierunek w ktorym probowano sie poruszyc byl inny przeciwnik.

#### 4.3.1.8 moveRandom()

```
void moveRandom (
            Enemy * enemy,
            tile ** bitmap,
            int gridSize )
```

Przesuwa przeciwnika typu 'random'.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.3.1.9 pointTowardsPlayerIfPossible()

```
int pointTowardsPlayerIfPossible (
            Enemy * enemy,
            Player * player )
```

Sprawdza, czy gracz jest w polu widzenia przeciwnika, tj sa w tej samej kolumnie lub wierszu i nie sa zaslonieci kolumna i nakierowuje na niego przeciwnika.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika, ktorego sprawdzane jest pole widzenia. |
| *player* | Wskaznik na gracza. |

**Zwraca**

Liczba 0 jesli nie widac gracza i nie zmieniono kierunku przeciwnika, liczba 1 jesli go widac i nakierowano na niego przeciwnika.

#### 4.3.1.10 resetEnemyStatics()

```
void resetEnemyStatics ( )
```

Wywoluje funkcje zawierajace zmienne statyczne tak, by je zresetowac.

## 4.4 Dokumentacja pliku Enemies.h

### Funkcje

- void initEnemies (Enemy enemyArray[MAX_ENEMIES_PER_MAP], tile *bitmap, int howMany, int gridSize)
- void resetEnemyStatics ()
- void addEnemy (Enemy enemyArray[MAX_ENEMIES_PER_MAP], int ID, EnemyType type, XY coords, Direction dir)
- void moveBlind (Enemy *enemy, tile **bitmap, int gridSize)
- void moveRandom (Enemy *enemy, tile **bitmap, int gridSize)
- int pointTowardsPlayerIfPossible (Enemy *enemy, Player *player)
- void moveCharger (Enemy *enemy, Player *player, tile **bitmap, int gridSize)
- int moveEnemy (Enemy *enemy, int rush, tile **bitmap, int gridSize)
- int manageEnemiesAndScore (Enemy enemyArray[MAX_ENEMIES_PER_MAP], Player *player, tile **bitmap, int howMany, int gridSize, int reset)
- void dropEnemies (Enemy enemyArray[MAX_ENEMIES_PER_MAP], tile **bitmap, int howMany, int grid↩ Size)

### 4.4.1 Dokumentacja funkcji

#### 4.4.1.1 addEnemy()

```
void addEnemy (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            int ID,
            EnemyType type,
            XY coords,
            Direction dir )
```

Dodaje nowego przeciwnika do tablicy i ustawia jego dane.

**Parametry**

| | |
|---|---|
| *enemyArray* | Tablica przechowujaca dane wszystkich przeciwnikow. |
| *ID* | Indeks nowego przeciwnika w tablicy przeciwnikow. |
| *type* | Typ przeciwnika. |
| *coords* | Koordynaty nowegp przeciwnika wzgledem mapy. |
| *dir* | Kiedunek poruszania sie nowego przeciwnika. |

#### 4.4.1.2 dropEnemies()

```
void dropEnemies (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            tile ** bitmap,
            int howMany,
            int gridSize )
```

Stawia nowych przeciwnikow na planszy.

**Parametry**

| | |
|---|---|
| *enemyArray* | Tablica przechowujaca dane wszystkich przeciwnikow. |
| *bitmap* | Wskaznik na mape gry. |
| *howMany* | Ile przeciwnikow jest do wygenerowania. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.4.1.3 initEnemies()

```
void initEnemies (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            tile * bitmap,
            int howMany,
            int gridSize )
```

Inicjalizuje rzeczy wymagajace inicjalizacji z tego pliku.

**Parametry**

| | |
|---|---|
| *enemyArray* | Tablica przechowujaca dane wszystkich przeciwnikow. |
| *bitmap* | Wskaznik na mape gry. |
| *howMany* | Ile przeciwnikow jest do wygenerowania. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.4.1.4 manageEnemiesAndScore()

```
int manageEnemiesAndScore (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            Player * player,
            tile ** bitmap,
            int howMany,
            int gridSize,
            int reset )
```

Obsluguje przeciwnikow, tj. przesuwa ich, sprawdza czy nie zostali zabici przez gracza (stoja na ogniu), dodaje punkty graczowi, jesli przeciwnik umiera i sprawdza, czy ktorys z przeciwnikow jeszcze zyje.

**Parametry**

| | |
|---|---|
| *enemyArray* | tablica przechowujaca dane wszystkich przeciwnikow. |
| *player* | Wskaznik na gracza. |
| *bitmap* | Wskaznik na mape gry. |
| *howMany* | Ile jest przeciwnikow. |
| *gridSize* | Wielkosc mapy (NxN). |
| *reset* | Jesli 1 to nalezy zresetowac wartosci statycze w funkcji i z niej wyjsc, bez wykonywania reszty jej ciala. |

**Zwraca**

Liczba 0 kiedy zyje przynajmniej jeden przeciwnik, liczba 1 jesli wszyscy sa martwi.

#### 4.4.1.5 moveBlind()

```
void moveBlind (
            Enemy * enemy,
            tile ** bitmap,
            int gridSize )
```

Przesuwa przeciwnika typu 'blind' lub przeciwnika typu 'charger' w pierwszej jego fazie.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.4.1.6 moveCharger()

```
void moveCharger (
            Enemy * enemy,
```

```
            Player * player,
            tile ** bitmap,
            int gridSize )
```

Przesuwa przeciwnika typu 'charger' w jego drugiej fazie.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *player* | Wskaznik na gracza. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.4.1.7 moveEnemy()

```
int moveEnemy (
            Enemy * enemy,
            int rush,
            tile ** bitmap,
            int gridSize )
```

Probuje przesunac przeciwnika w jego kierunku o jeden kafelek.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *rush* | Przelacznik, czy przeciwnik ma sie przebijac przez pudelka. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

**Zwraca**

Liczba 0 jesli sie nie udalo przesunac przeciwnika, liczba 1 w przypadku kiedy sie udalo, 2 jesli kierunek w ktorym probowano sie poruszyc byl inny przeciwnik.

#### 4.4.1.8 moveRandom()

```
void moveRandom (
            Enemy * enemy,
            tile ** bitmap,
            int gridSize )
```

Przesuwa przeciwnika typu 'random'.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika do przesuniecia. |
| *bitmap* | Wskaznik na mape gry. |
| *gridSize* | Wielkosc mapy (NxN). |

#### 4.4.1.9 pointTowardsPlayerIfPossible()

```
int pointTowardsPlayerIfPossible (
            Enemy * enemy,
            Player * player )
```

Sprawdza, czy gracz jest w polu widzenia przeciwnika, tj sa w tej samej kolumnie lub wierszu i nie sa zaslonieci kolumna i nakierowuje na niego przeciwnika.

**Parametry**

| | |
|---|---|
| *enemy* | Wskaznik na przeciwnika, ktorego sprawdzane jest pole widzenia. |
| *player* | Wskaznik na gracza. |

**Zwraca**

Liczba 0 jesli nie widac gracza i nie zmieniono kierunku przeciwnika, liczba 1 jesli go widac i nakierowano na niego przeciwnika.

#### 4.4.1.10 resetEnemyStatics()

```
void resetEnemyStatics ( )
```

Wywoluje funkcje zawierajace zmienne statyczne tak, by je zresetowac.

## 4.5 Dokumentacja pliku enumsStructsMacros.h

### Struktury danych

- struct XY
- struct tile
- struct Bomb
- struct Enemy
- struct Player

### Definicje

- #define **DEFAULT_GRID_SIZE** 11
- #define **MAX_GRID_SIZE** 100
- #define **MAX_ENEMIES_PER_MAP** 20
- #define **DEFAULT_ENEMIES_COUNT** 4
- #define **DEFAULT_GAME_TIME** 180.9999
- #define **DEFAULT_DEBUG_MODE** 0
- #define **MAX_BOMBS** 20
- #define **INIT_BOMB_AVALIBLE** 1
- #define **INIT_BOMB_KICK** 0
- #define **INIT_SHIELD** 0
- #define **INIT_FIRERANGE** 2
- #define **BOX_COLOR** 0.5, 0.5, 0.5
- #define **PATH_COLOR** 0.1, 0.1, 0.1
- #define **PILLAR_COLOR** 1,1,1
- #define **FIRE_COLOR** 0.9, 0.7, 0.1
- #define **FIRED_BOX_COLOR** 0.4, 0.15, 0.0
- #define **PORTAL_COLOR** 0.3, 0.3, 1
- #define **PORTAL_COLOR_2** 0.7, 0.7, 1
- #define **PORTAL_COLOR_3** 0.9, 0.9, 1
- #define **PLAYER_SIZE** 0.91
- #define **PLAYER_COLOR** 0.3, 0.55, 1.0
- #define **SHIELD_COLOR** 0.6, 0.9, 1
- #define **DEAD_PLAYER_COLOR** 0.5, 0.1, 0.1
- #define **UPGRADE_CRYSTAL_SIZE** 0.9
- #define **UPG_FIRE_RANGE_COLOR** FIRE_COLOR
- #define **UPG_SHIELD_COLOR** SHIELD_COLOR
- #define **UPG_BOMB_KICK_COLOR** 1, 0.7, 0.7
- #define **UPG_BOMBS_AVALIBLE** 0.8, 0.1, 0.1
- #define **SCOREBOARD_BACKGROUND_COLOR** PATH_COLOR
- #define **BACKGROUND_COLOR** PILLAR_COLOR
- #define **FONT_COLOR** 1,1,1
- #define **FONT_SIZE** 0.01
- #define **PAUSE_SCREEN_COLOR** 0.1,0.1,0.1
- #define **GRID_L_X** -0.965
- #define **GRID_D_Y** -0.95
- #define **SCORE_BOARD_L_X** 1.05/3.0
- #define **SCORE_BOARD_R_X** 0.95
- #define **SCORE_BOARD_D_Y** -0.95
- #define **SCORE_BOARD_U_Y** 0.95
- #define **PLAYER_APPEAR_TIME** 1.5
- #define **BROKEN_SHIELD_TIME** 1.5
- #define **FIRE_LINGERING_TIME** 0.5
- #define **BOMB_EXPLODE_TIME** 2.5
- #define **SCORE_DROP_FREQUENCY** 1
- #define **TIME_COUNT_SPEED** 0.07
- #define **PORTAL_SUCKING_SPEED** 0.75
- #define **ENEMY1_SPEED** 0.8
- #define **ENEMY2_SPEED** 0.5
- #define **ENEMY3_SPEED** 0.2
- #define **RANDOM_TRIALS** 2
- #define **ENEMY1_COLOR** 0.9, 0.9, 0.0
- #define **ENEMY1_DEAD_COLOR** DEAD_PLAYER_COLOR
- #define **ENEMY2_COLOR** 0.0, 0.8, 0.8

- #define **ENEMY2_DEAD_COLOR** DEAD_PLAYER_COLOR
- #define **ENEMY3_1ST_PHASE_COLOR** 0.3, 0.0, 0.3
- #define **ENEMY3_2ND_PHASE_COLOR** 0.9, 0.0, 0.9
- #define **ENEMY3_DEAD_COLOR** DEAD_PLAYER_COLOR
- #define **ENEMY_ROT_TIME** 0.7

## Wyliczenia

- enum Direction { **north** = 1, **east**, **south**, **west** }
- enum upgrades { fireRange = 1, bombsAvalible, bombKick, shield }
- enum tileType {
  pathTile, boxTile, pillarTile, bombTile,
  fireTile, firedBox, portalTile, enemyTile }
- enum bombState {
  empty, placed, exploding, exploded,
  toBeCleaned }
- enum PauseScreenType {
  none, start, stop, gameOver,
  won }
- enum EnemyType { blind, random, charger }
- enum Status {
  dead, alive, winning, dying,
  rushing, calm, appearing }

### 4.5.1 Dokumentacja typów wyliczanych

#### 4.5.1.1 bombState

`enum bombState`

Status bomby

**Wartości wyliczeń**

| | |
|---|---|
| empty | Bomba nie jest polozona. |
| placed | Bomba jest polozona i jeszcze nie wybuchla. |
| exploding | Bomba wlasnie wybuchnela. |
| exploded | Bomba jest po wybuchu. |
| toBeCleaned | Bomba oczekuje na likwidacje wraz z jej ogniem. |

#### 4.5.1.2 Direction

`enum Direction`

Mozliwe kierunki poruszania sie

#### 4.5.1.3 EnemyType

`enum EnemyType`

Typy przeciwnikow

**Wartości wyliczeń**

| | |
|---|---|
| blind | Porusza sie po prostej az do napotkania przeszkody (kafelek 'boxTile', 'pillarTile', 'firedBox', 'enemytile'), po czym probuje zmienic kierunek. Najpierw w lewo wzgledem pierwotnego kierunku, jesli nie da sie poruszyc w nowym kierunku, to probuje poruszyc sie w prawo wzgledem oryginalnego kierunku, jak i to sie nie uda, to zawraca wzgledem oryginalnego kierunku. Jesli nie moze sie poruszyc w zadnym z kierunkow, to sie nie porusza. |
| random | Porusza sie w przypadkowych kierunkach. Jesli nie moze sie poruszyc w danym kierunku, to probuje kolejny przypadkowy kierunek okreslona makrem RANDOM_TRIALS ilosc razy. |
| charger | Porusza sie tak jak przeciwnik typu 'blind' az w jego polu widzenia znajdzie sie gracz, tj bedzie w tej samej kolumnie lub wierszu i jesli w linii prostej miedzy graczem i przeciwnikiem nie bedzie kafelka typu 'pillarTile', wtedy zmienia swoj tryb i porusza sie po linii prostej w kierunku w jakim ostatni raz 'widzial' gracza az do napotkania konca mapy lub innego przeciwnika, wtedy, jesli nie 'widzi' gracza, to wraca do trybu 'calm' i znow porusza sie jak przeciwnik typu 'blind'. |

#### 4.5.1.4 PauseScreenType

`enum PauseScreenType`

Typy pauzy

**Wartości wyliczeń**

| | |
|---|---|
| none | Aktualnie nie ma pauzy. |
| start | Pauza w oczekiwaniu na rozpoczecie gry. |
| stop | Pauza w trakcie gry. |
| gameOver | Pauza po przegraniu planszy. |
| won | Pauza po wygraniu planszy. |

#### 4.5.1.5 Status

`enum Status`

Mozliwy status gracza lub przeciwnika

**Wartości wyliczeń**

| | |
|---|---|
| dead | Status martwego gracza oraz przeciwnika, ktory umarl chwile temu i minelo dosc duzo czasu, zeby wylaczyc rysowanie jego korpusu. |

**Wartości wyliczeń**

| | |
|---|---|
| alive | Status zywego gracza i przeciwnika. |
| winning | Status gracza jesli wejdzie do portalu. |
| dying | Status przeciwnika, ktory przed chwila umarl i jeszcze nie minelo dosc duzo czasu, zeby wylaczyc rysowanie jego korpusu. |
| rushing | Status przeciwnika typu 'charger', jesli 'zobaczyl' (sposob 'widzenia' opisany w komentarzu w do enumeratora EnrmyType) gracza. |
| calm | Status przeciwnika typu 'charger' jesli nie 'widzial' (sposob 'widzenia' opisany w komentarzu w do enumeratora EnrmyType) przez jakis czas gracza. |
| appearing | Status gracza pojawiajacego sie na mapie (pierwszy moment gry). |

#### 4.5.1.6 tileType

`enum tileType`

Typy kafelek mapy

**Wartości wyliczeń**

| | |
|---|---|
| pathTile | Kafelek sciezki - gracz, przeciwnik oraz przesuwajaca sie bomba moga sie po nim poruszac. |
| boxTile | Kafelek pudelka - wybuch bomby moze go zniszczyc, poruszajaca bomba sie przed nim zatrzyma, nic nie moze przez niego przejsc, poza przeciwnikiem typu 'charger' w jego drugiej fazie. |
| pillarTile | Kafelek kolumny - nic nie moze przez niego przejsc, nie mozna go w zaden sposob zniszczyc, przeciwnik typu 'charger' nie zobaczy przeciwnika, jesli na jego polu widzenia znajduje sie ten kafelek. |
| bombTile | Kafelek na ktorym znajduje sie bomba - nic przez niego nie moze przejsc, poza graczem, ktory ma ulepszenie 'bombKick', w takim wypadku gracz moze wejsc na ten kafelek, a bomba na nim sie znajdujaca zacznie sie przesuwac. |
| fireTile | Kafelek na ktorym znajduje sie ogien - wszystko moze na niego wejsc, przy czym przeciwnicy i gracz bez ulepszenia 'shield' znajdujacy sie na tym kafelku umieraja. Gracz posiadajacy ulepszenie shield straci je. |
| firedBox | Kafelek na ktorym znajduje sie pudelko, ktore dosiegnela eksplozja bomby - przeciwnicy nie moga przez niego przejsc poza przeciwnikiem typu 'charger' w jego drugiej fazie, ktory wchodzac na niego umrze. Dla gracza zachowuje sie jak kafelek 'fireTile'. |
| portalTile | Kafelek, ktory pojawia sie na srodku mapy w momencie, kiedy wszyscy przeciwnicy zostana pokonani. Kiedy gracz na niego wejdzie, to rozpocznie podsumowanie punktow i pauza typu zwyciestwo. |
| enemyTile | Kafelek na ktorym znajduje sie przeciwnik - inni przeciwnicy nie moga na niego wejsc, gracz wchodzac na nie otrzymuje obrazenia. Nie mozna na nim postawic bomby, a poruszajaca bomba sie przed nim zatrzyma. |

#### 4.5.1.7 upgrades

`enum upgrades`

Typy ulepszen

**Wartości wyliczeń**

| | |
|---|---|
| fireRange | Zasieg bomby. |
| bombsAvalible | Ile bomb na raz moze byc postawionych na mapie. |
| bombKick | Czy gracz moze "kopnac" bombe, tj sprawic, zeby bomba przesuwala sie w okreslonym kierunku. |
| shield | Czy gracz ma tarcze. Jesli tak, to moze stanac na ogniu albo spotkac przeciwnika bez natychmiastowej smierci, ale wtedy tarcza w ciagu chwili zniknie. |

## 4.6 Dokumentacja pliku fonts.h

### Definicje typów

- typedef int(∗ sign)[8][5]

### Funkcje

- void translateWord (sign ∗∗signs, char ∗word)
- void writeStaticData ()
- void writeDynamicData (int highscore, int score, int timeLeft)
- void writePauseMessage (PauseScreenType pauseType)
- void writePauseMessageScore (int score)

### 4.6.1 Dokumentacja definicji typów

#### 4.6.1.1 sign

```
typedef int(* sign)[8][5]
```

Wskaznik na tablice reprezentujaca znak w postaci bitow - pixeli

### 4.6.2 Dokumentacja funkcji

#### 4.6.2.1 translateWord()

```
void translateWord (
            sign ** signs,
            char * word )
```

Zamienia wejsciowy ciag znakow, ta tablice wskaznikow na tablice reprezentujace znaki w postaci bitow - pixeli

**Parametry**

| | | |
|---|---|---|
| out | *signs* | wskaznik na wskaznik na tablice wskaznikow na tablice reprezentujace znaki w postaci bitow - pixeli |
| | *word* | ciag znakow do przekonwertowania |

#### 4.6.2.2 writeDynamicData()

```
void writeDynamicData (
            int highscore,
            int score,
            int timeLeft )
```

Zapisuje w obszarze okna obok mapy aktualna ilosc punktow, dotychczasowa najwieksza ilosc punktow oraz pozostaly czas gracza pod odpowiednimi tytulami

**Parametry**

| | |
|---|---|
| *highscore* | najwyzsza dotychczasowa ilosc punktow |
| *score* | aktualna ilosc punktow |
| *timeLeft* | pozostaly czas gracza |

#### 4.6.2.3 writePauseMessage()

```
void writePauseMessage (
            PauseScreenType pauseType )
```

Wypisuje na okno pauzy wiadomosc, zalezaca od typu pauzy

**Parametry**

| | |
|---|---|
| *pauseType* | typ pauzy |

#### 4.6.2.4 writePauseMessageScore()

```
void writePauseMessageScore (
            int score )
```

Wypisuje na okno pauzy aktualna ilosc punktow

**Parametry**

| | |
|---|---|
| *score* | aktualna ilosc punktow |

#### 4.6.2.5 writeStaticData()

```
void writeStaticData ( )
```

Zapisuje w obszarze okna obok mapy tytuly informacji: aktualne punkty, najwieksza ilosc punktow oraz pozostaly czas gracza

## 4.7 Dokumentacja pliku highscoreFile.c

```
#include <stdio.h>
```

### Funkcje

- int getHighscore (FILE ∗in)
- void setHighscore (int score, FILE ∗out)

### 4.7.1 Dokumentacja funkcji

#### 4.7.1.1 getHighscore()

```
int getHighscore (
            FILE * in )
```

Pobiera z pliku wartosc dotychczasowej najwiekszej ilosci punktow

**Parametry**

| | | |
|---|---|---|
| | *in* | wskaznik na strukture pliku |
| out | *pobrana* | z pliku wartosc dotychczasowej najwiekszej ilosci punktow |

#### 4.7.1.2 setHighscore()

```
void setHighscore (
```

```
            int score,
            FILE * out )
```

Zapisuje do pliku wartosc dotychczasowej najwiekszej ilosci punktow

**Parametry**

| | |
|---|---|
| *out* | wskaznik na strukture pliku |
| *score* | wartosc dotychczasowej najwiekszej ilosci punktow |

## 4.8 Dokumentacja pliku highscoreFile.h

### Funkcje

- int getHighscore (FILE ∗in)
- void setHighscore (int score, FILE ∗out)

### 4.8.1 Dokumentacja funkcji

#### 4.8.1.1 getHighscore()

```
int getHighscore (
            FILE * in )
```

Pobiera z pliku wartosc dotychczasowej najwiekszej ilosci punktow

**Parametry**

| | | |
|---|---|---|
| | *in* | wskaznik na strukture pliku |
| out | *pobrana* | z pliku wartosc dotychczasowej najwiekszej ilosci punktow |

#### 4.8.1.2 setHighscore()

```
void setHighscore (
            int score,
            FILE * out )
```

Zapisuje do pliku wartosc dotychczasowej najwiekszej ilosci punktow

**Parametry**

| | |
|---|---|
| *out* | wskaznik na strukture pliku |
| *score* | wartosc dotychczasowej najwiekszej ilosci punktow |

## 4.9 Dokumentacja pliku main.c

```
#include <GLFW\glfw3.h>
#include <stdio.h>
#include "enumsStructsMacros.h"
#include "draw.h"
#include "player.h"
#include "manage bombs.h"
#include "Enemies.h"
#include "fonts.h"
#include "highscoreFile.h"
#include "TestArg.h"
#include "mapMemAndGen.h"
```

### Funkcje

- void pauseGame (PauseScreenType *pauseSetter, double *dTimeSetter)
- int unPauseGame (PauseScreenType *pauseSetter, double *dTimeSetter)
- static void key_callback (GLFWwindow *window, int key, int scancode, int action, int mods, tile ***bitmap↩ Setter, Player *playerSetter, int *debugModeSetter, int *gridSizeSetter)
- void error_callback (int error, const char *description)
- void setDefaultsIfNecessary (int *howManyEnemies, int *howManyBoxes, int *howManyUpgrades, int *gridSize, int *gameTime, int *debugMode)
- void init (Enemy enemyArray[MAX_ENEMIES_PER_MAP], tile **(*bitmap), PauseScreenType *pauseType, double *dTime, Player *player, int howManyEnemies, int howManyBoxes, int howManyUpgrades, int *grid↩ Size, int gameTime, int *debugMode, int listArray[6])
- void resetTimes (double *dTime)
- int **main** (int argc, char **argv)

### 4.9.1 Dokumentacja funkcji

#### 4.9.1.1 error_callback()

```
void error_callback (
            int error,
            const char * description )
```

Wypisuje do konsoli blad.

**Parametry**

| | |
|---|---|
| *error* | numer bledu |
| *description* | opis bledu |

#### 4.9.1.2 init()

```
void init (
            Enemy enemyArray[MAX_ENEMIES_PER_MAP],
            tile *** bitmap,
            PauseScreenType * pauseType,
            double * dTime,
            Player * player,
            int howManyEnemies,
            int howManyBoxes,
            int howManyUpgrades,
            int * gridSize,
            int gameTime,
            int * debugMode,
            int listArray[6] )
```

Inicjalizuje gre oraz ustawia statyczne zmienne w konkretnych funkcjach

**Parametry**

| | |
|---|---|
| *enemy* | wskaznik na przeciwnika do przesuniecia |
| *bitmap* | wskaznik na wskaznik na mape gry |
| *pauseType* | Wskaznik na typ pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji. |
| *dTime* | Wskaznik na zmienna pamietajaca czas gry sprzed pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji. |
| *player* | wskaznik na gracza |
| *howManyEnemies* | Ile ma sie znajdowac przeciwnikow na mapie. |
| *howManyBoxes* | Ile ma sie znajdowac pudelek na mapie. |
| *howManyUpgrades* | Ile ma sie znajdowac ulepszen na mapie. |
| *gridSize* | wielkosc mapy |
| *gameTime* | czas gry |
| *debugMode* | zmienna ustawiajaca debugMode gry |
| *[out* | listArray Tablica przechowujaca indeksy glList. |

#### 4.9.1.3 key_callback()

```
static void key_callback (
            GLFWwindow * window,
            int key,
            int scancode,
            int action,
            int mods,
            tile *** bitmapSetter,
            Player * playerSetter,
            int * debugModeSetter,
            int * gridSizeSetter )  [static]
```

Wywoluje odpowiednie funkcje odpowiadajace wcisnieciu odpowiednich klawiszy

**Parametry**

| | |
|---|---|
| *window* | wskaznik na strukture okna glfw. |
| *key* | Jaki klawisz. |
| *scancode* | |
| *action* | Akcja klawisza (nacisniety czy puszczony). |
| *mods* | |
| *bitmap* | wskaznik na wskaznik na mape gry. Sluzy do ustawienia zmiennych statycznych wewnatrz funkcji) |
| *playerSetter* | wskaznik gracza. Sluzy do ustawienia zmiennych statycznych wewnatrz funkcji) |
| *debugModeSetter* | wskaznik zmienna ustawiajaca debugMode gry. Sluzy do ustawienia zmiennych statycznych wewnatrz funkcji) |
| *gridSizeSetter* | wskaznik zmienna przechowujaca wielkosc mapy. Sluzy do ustawienia zmiennych statycznych wewnatrz funkcji) |

#### 4.9.1.4 pauseGame()

```
void pauseGame (
            PauseScreenType * pauseSetter,
            double * dTimeSetter )
```

Pauzuje gre i zapisuje aktualny czas

**Parametry**

| | |
|---|---|
| *pauseSetter* | Wskaznik na typ pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji. |
| *dTimeSetter* | Wskaznik na zmienna pamietajaca czas gry sprzed pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji. |

#### 4.9.1.5 resetTimes()

```
void resetTimes (
            double * dTime )
```

Wywoluje funkcje resetujace wszystkie funkcje statyczne oraz resetuje czas

**Parametry**

| | | |
|---|---|---|
| out | *dTime* | wskaznik na aktualny czas gry |

#### 4.9.1.6 setDefaultsIfNecessary()

```
void setDefaultsIfNecessary (
            int * howManyEnemies,
            int * howManyBoxes,
            int * howManyUpgrades,
            int * gridSize,
            int * gameTime,
            int * debugMode )
```

Ustawia wartosci domyslne przekazanym zmiennym, jesli nie zostaly ustawione.

**Parametry**

| | | |
|---|---|---|
| out | *howManyEnemies* | Ile ma sie znajdowac przeciwnikow na mapie. |
| out | *howManyBoxes* | Ile ma sie znajdowac pudelek na mapie. |
| out | *howManyUpgrades* | Ile ma sie znajdowac ulepszen na mapie. |
| out | *gridSize* | wielkosc mapy |
| out | *gameTime* | czas gry |
| out | *debugMode* | zmienna ustawiajaca debugMode gry |

#### 4.9.1.7 unPauseGame()

```
int unPauseGame (
            PauseScreenType * pauseSetter,
            double * dTimeSetter )
```

Wylacza pauze gry i ustawia aktualny czas na czas sprzed pauzy

**Parametry**

| | |
|---|---|
| *pauseSetter* | Wskaznik na typ pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji. |
| *dTimeSetter* | Wskaznik na zmienna pamietajaca czas gry sprzed pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji. |

**Zwraca**

1 jesli udalo sie wylaczyc pauze lub 0 jesli pauzy nie bylo

## 4.10 Dokumentacja pliku manage bombs.C

```
#include "enumsStructsMacros.h"
#include "manage bombs.h"
#include <stdlib.h>
#include <GLFW\glfw3.h>
```

## Funkcje

- int getBombID (Player *player, int x, int y)
- void moveTile (enum Direction dir, Bomb *currBomb, tile **bitmap, int gridSize)
- void moveCheckExplodeAndCleanUpBombs (Player *player, tile **bitmap, int gridSize)
- void cleanTile (tile *currTile)
- void moveBombData (Bomb *A, Bomb *B)
- void cleanUp (int currentBombID, Player *player, tile **bitmap)
- void spreadFire (tile *currTile, int *breaker)
- void explodeBombAddPoints (Bomb *bomb, Player *player, tile **bitmap, int gridSize)

### 4.10.1 Dokumentacja funkcji

#### 4.10.1.1 cleanTile()

```
void cleanTile (
            tile * currTile )  [inline]
```

Zmniejsza ilosc warstw ognia na kafelku i zamienia go na kafelek typu 'pathTile', jesli ilosc warstw ognia sie wyzeruje.

**Parametry**

| | |
|---|---|
| *currTile* | Kafelek do obsluzenia. |

#### 4.10.1.2 cleanUp()

```
void cleanUp (
            int currentBombID,
            Player * player,
            tile ** bitmap )
```

Usuwa ogien pozostaly po eksplozji bomby, jesli minelo dosc duzo czasu od jej wybuchu

**Parametry**

| | |
|---|---|
| *currentBombID* | indeks w tablicy bomb gracza aktualnej bomby |
| *player* | wskaznik na gracza |
| *bitmap* | wskaznik na mape gry |

#### 4.10.1.3 explodeBombAddPoints()

```
void explodeBombAddPoints (
            Bomb * bomb,
            Player * player,
            tile ** bitmap,
            int gridSize )
```

Zmienia status bomby na 'exploded', roznosi ogien po mapie zgodnie z zasiegiem danej bomby zwieksza ilosc wolnych bomb gracza

**Parametry**

| | |
|---|---|
| *bomb* | wskaznik na bombe, ktora nalezy eksplodowac |
| *player* | wskaznik na gracza |
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.10.1.4 getBombID()

```
int getBombID (
            Player * player,
            int x,
            int y )
```

Znajduje ID w tablicy bomb gracza bomby o okreslonych koordynatach

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *x* | koordynat szerokosci wzgledem mapy |
| *y* | koordynat wysokosci wzgledem mapy |

**Zwraca**

indeks, pod ktorym w tablicy bomb gracza znajduje sie bomba o podanych koordynatach

#### 4.10.1.5 moveBombData()

```
void moveBombData (
            Bomb * A,
            Bomb * B )  [inline]
```

Przekazuje dane bomby A do bomby B.

**Parametry**

| | |
|---|---|
| *A* | bomba, ktorej dane sa przekazywane |
| *B* | bomba, do ktorej dane sa przekazywane |

#### 4.10.1.6 moveCheckExplodeAndCleanUpBombs()

```
void moveCheckExplodeAndCleanUpBombs (
            Player * player,
            tile ** bitmap,
            int gridSize )
```

Obsluguje bomby - przesuwa je, sprawdza, czy ktoras nie stoi na ogniu i nie musi wybuchnac, czysci kafelki po bombach, ktore wybuchly i minelo dosc duzo czasu od ich wybuchu

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.10.1.7 moveTile()

```
void moveTile (
            enum Direction dir,
            Bomb * currBomb,
            tile ** bitmap,
            int gridSize )
```

Jesli moze, to przesuwa bombe o jeden kafelek

**Parametry**

| | |
|---|---|
| *dir* | kierunek poruszania sie bomby |
| *currBomb* | wskaznik na aktualnie przesuwana bombe |
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.10.1.8 spreadFire()

```
void spreadFire (
            tile * currTile,
            int * breaker )  [inline]
```

Probuje dany kafelek zamienic na 'fireTile' lub 'firedBox', jesli sie uda zamienic na 'fireTile', to zwraca ∗breaker = 0, czyli ogien moze isc dalej, jesli sie uda zamienic na 'firedBox', to zwraca ∗breaker = 1, czyli ogien nie isc dalej, jesli sie nie uda, to zwraca ∗breaker = 2, czyli ogien nie siegnal do tego kafelka i nie moze isc dalej

**Parametry**

| | | |
|---|---|---|
| | *currTile* | wskaznik na aktualny kafelek do sprawdzenia |
| out | *breaker* | ustawia jego wartosc na 1 jesli dosiegnie pudelka |

## 4.11 Dokumentacja pliku manage bombs.h

### Funkcje

- int getBombID (Player ∗player, int x, int y)
- void moveTile (enum Direction dir, Bomb ∗currBomb, tile ∗∗bitmap, int gridSize)
- void moveCheckExplodeAndCleanUpBombs (Player ∗player, tile ∗∗bitmap, int gridSize)
- void cleanTile (tile ∗currTile)
- void moveBombData (Bomb ∗A, Bomb ∗B)
- void cleanUp (int currentBombID, Player ∗player, tile ∗∗bitmap)
- void spreadFire (tile ∗currTile, int ∗breaker)
- void explodeBombAddPoints (Bomb ∗bomb, Player ∗player, tile ∗∗bitmap, int gridSize)

### 4.11.1 Dokumentacja funkcji

#### 4.11.1.1 cleanTile()

```
void cleanTile (
            tile * currTile )  [inline]
```

Zmniejsza ilosc warstw ognia na kafelku i zamienia go na kafelek typu 'pathTile', jesli ilosc warstw ognia sie wyzeruje.

**Parametry**

| | |
|---|---|
| *currTile* | Kafelek do obsluzenia. |

#### 4.11.1.2 cleanUp()

```
void cleanUp (
            int currentBombID,
            Player * player,
            tile ** bitmap )
```

Usuwa ogien pozostaly po eksplozji bomby, jesli minelo dosc duzo czasu od jej wybuchu

**Parametry**

| | |
|---|---|
| *currentBombID* | indeks w tablicy bomb gracza aktualnej bomby |
| *player* | wskaznik na gracza |
| *bitmap* | wskaznik na mape gry |

#### 4.11.1.3 explodeBombAddPoints()

```
void explodeBombAddPoints (
            Bomb * bomb,
            Player * player,
            tile ** bitmap,
            int gridSize )
```

Zmienia status bomby na 'exploded', roznosi ogien po mapie zgodnie z zasiegiem danej bomby zwieksza ilosc wolnych bomb gracza

**Parametry**

| | |
|---|---|
| *bomb* | wskaznik na bombe, ktora nalezy eksplodowac |
| *player* | wskaznik na gracza |
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.11.1.4 getBombID()

```
int getBombID (
            Player * player,
            int x,
            int y )
```

Znajduje ID w tablicy bomb gracza bomby o okreslonych koordynatach

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *x* | koordynat szerokosci wzgledem mapy |
| *y* | koordynat wysokosci wzgledem mapy |

**Zwraca**

indeks, pod ktorym w tablicy bomb gracza znajduje sie bomba o podanych koordynatach

#### 4.11.1.5 moveBombData()

```
void moveBombData (
            Bomb * A,
            Bomb * B )  [inline]
```

Przekazuje dane bomby A do bomby B.

**Parametry**

| | |
|---|---|
| *A* | bomba, ktorej dane sa przekazywane |
| *B* | bomba, do ktorej dane sa przekazywane |

#### 4.11.1.6 moveCheckExplodeAndCleanUpBombs()

```
void moveCheckExplodeAndCleanUpBombs (
            Player * player,
            tile ** bitmap,
            int gridSize )
```

Obsluguje bomby - przesuwa je, sprawdza, czy ktoras nie stoi na ogniu i nie musi wybuchnac, czysci kafelki po bombach, ktore wybuchly i minelo dosc duzo czasu od ich wybuchu

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.11.1.7 moveTile()

```
void moveTile (
            enum Direction dir,
            Bomb * currBomb,
            tile ** bitmap,
            int gridSize )
```

Jesli moze, to przesuwa bombe o jeden kafelek

**Parametry**

| | |
|---|---|
| *dir* | kierunek poruszania sie bomby |
| *currBomb* | wskaznik na aktualnie przesuwana bombe |
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.11.1.8 spreadFire()

```
void spreadFire (
            tile * currTile,
            int * breaker )  [inline]
```

Probuje dany kafelek zamienic na 'fireTile' lub 'firedBox', jesli sie uda zamienic na 'fireTile', to zwraca ∗breaker = 0, czyli ogien moze isc dalej, jesli sie uda zamienic na 'firedBox', to zwraca ∗breaker = 1, czyli ogien nie isc dalej, jesli sie nie uda, to zwraca ∗breaker = 2, czyli ogien nie siegnal do tego kafelka i nie moze isc dalej

**Parametry**

| | | |
|---|---|---|
| | *currTile* | wskaznik na aktualny kafelek do sprawdzenia |
| out | *breaker* | ustawia jego wartosc na 1 jesli dosiegnie pudelka |

## 4.12 Dokumentacja pliku mapMemAndGen.c

```
#include "enumsStructsMacros.h"
#include "mapMemAndGen.h"
#include <stdio.h>
```

### Funkcje

- void dealocateMap (tile ∗∗(∗bitmap), int gridSize)
- void allocateMap (tile ∗∗(∗bitmap), int gridSize)
- void generateMap (tile ∗∗bitmap, int gridSize, int howManyBoxes, int howManyUpgrades)

### 4.12.1 Dokumentacja funkcji

#### 4.12.1.1 allocateMap()

```
void allocateMap (
            tile *** bitmap,
            int gridSize )
```

Alokuje pamiec dla mapy o okreslonej wielkosci.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.12.1.2 dealocateMap()

```
void dealocateMap (
            tile *** bitmap,
            int gridSize )
```

Dealokuje pamiec mapy o okreslonej wielkosci.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.12.1.3 generateMap()

```
void generateMap (
            tile ** bitmap,
            int gridSize,
            int howManyBoxes,
            int howManyUpgrades )
```

Generuje mape gry - stawia kolumny, pudelka i ulepszenia

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy (NxN) |
| *howManyBoxes* | ile na mapie ma sie znajdowac sie pudel |
| *howManyUpgrades* | ile na mapie ma sie znajdowac ulepszen |

## 4.13 Dokumentacja pliku mapMemAndGen.h

### Funkcje

- void dealocateMap (tile $**$(*bitmap), int gridSize)
- void allocateMap (tile $**$(*bitmap), int gridSize)
- void generateMap (tile $**$bitmap, int gridSize, int howManyBoxes, int howManyUpgrades)

### 4.13.1 Dokumentacja funkcji

#### 4.13.1.1 allocateMap()

```
void allocateMap (
            tile *** bitmap,
            int gridSize )
```

Alokuje pamiec dla mapy o okreslonej wielkosci.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.13.1.2 dealocateMap()

```
void dealocateMap (
            tile *** bitmap,
            int gridSize )
```

Dealokuje pamiec mapy o okreslonej wielkosci.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.13.1.3 generateMap()

```
void generateMap (
            tile ** bitmap,
            int gridSize,
            int howManyBoxes,
            int howManyUpgrades )
```

Generuje mape gry - stawia kolumny, pudelka i ulepszenia

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy (NxN) |
| *howManyBoxes* | ile na mapie ma sie znajdowac sie pudel |
| *howManyUpgrades* | ile na mapie ma sie znajdowac ulepszen |

## 4.14 Dokumentacja pliku player.c

```
#include "enumsStructsMacros.h"
#include "player.h"
#include <stdlib.h>
#include <GLFW\glfw3.h>
```

### Funkcje

- void moveInADirectionAndKickIfPossibleAndCollectAnUpgrade (Direction dir, tile **bitmap, Player *player, int gridSize)
- void initPlayer (Player *player, int gameTime, int gridSize)
- void resetPlayerStatics ()
- void reInitPlayer (Player *player, int gameTime, int gridSize)
- void openPortal (tile **bitmap, int gridSize)
- void placeBomb (tile **bitmap, Player *player, int gridSize)
- void placeBox (tile **bitmap, Player *player, int gridSize)
- void damagePlayer (Player *player)
- void checkPlayer (tile **bitmap, Player *player, int gameTime, int reset)
- int waitForGame_over ()
- int waitForNextStageAndSumScore (Player *player)
- void increaseBombFireRange (Player *player)
- void decreaseBombFireRange (Player *player)
- void increaseBombsAvalible (Player *player)
- void decreaseBombsAvalible (Player *player)
- void shieldSwitch (Player *player)
- void kickSwitch (Player *player)

### 4.14.1 Dokumentacja funkcji

#### 4.14.1.1 checkPlayer()

```
void checkPlayer (
            tile ** bitmap,
            Player * player,
            int gameTime,
            int reset )
```

Stawia pudelko na kafelku na ktorym stoi gracz. Funkcja dostepna tylko w trybie debugu.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gameTime* | czas, w ktorym gracz ma skonczyc plansze |
| *reset* | jesli 1 to nalezy zresetowac wartosci statycze w funkcji i z niej wyjsc, bez wykonywania reszty jej ciala |

#### 4.14.1.2 damagePlayer()

```
void damagePlayer (
            Player * player )
```

Zabija gracza, jesli ten nie ma aktywnego ulepszenia tarczy. Jesli ma, to po chwili sie je wylacza.

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.3 decreaseBombFireRange()

```
void decreaseBombFireRange (
            Player * player )
```

Obniza zasieg bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.4 decreaseBombsAvalible()

```
void decreaseBombsAvalible (
            Player * player )
```

Zmniejsza ilosc dostepnych bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.5 increaseBombFireRange()

```
void increaseBombFireRange (
            Player * player )
```

Zwieksza zasieg bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.6 increaseBombsAvalible()

```
void increaseBombsAvalible (
            Player * player )
```

Zwieksza ilosc dostepnych bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.7 initPlayer()

```
void initPlayer (
            Player * player,
            int gameTime,
            int gridSize )
```

Inicjalizuje gracza, tj. ustawia mu wszystkie dane

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *gameTime* | czas, w ktorym gracz ma skonczyc plansze |
| *gridSize* | wielkosc mapy |

#### 4.14.1.8 kickSwitch()

```
void kickSwitch (
            Player * player )
```

Wlacza lub wylacza ulepszenie kopania bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.9 moveInADirectionAndKickIfPossibleAndCollectAnUpgrade()

```
void moveInADirectionAndKickIfPossibleAndCollectAnUpgrade (
            Direction dir,
            tile ** bitmap,
            Player * player,
            int gridSize )
```

Przesuwa gracza, jesli moze. Jesli na nowym kafelku bylo ulepszenie, to je zbiera, a jesli na nowym kafelku byla bomba, a gracz ma ulepszenie kopania bomb, to bomba ta zostanie kopnieta

**Parametry**

| | |
|---|---|
| *dir* | kierunek przesuniecia gracza |
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy |

#### 4.14.1.10 openPortal()

```
void openPortal (
            tile ** bitmap,
            int gridSize )
```

Otwiera portal, czyli zamienia srodkowy kafelek mapy na kafelek portalu

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.14.1.11 placeBomb()

```
void placeBomb (
            tile ** bitmap,
            Player * player,
            int gridSize )
```

Stawia bombe na kafelku na ktorym stoi gracz

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy |

#### 4.14.1.12 placeBox()

```
void placeBox (
            tile ** bitmap,
            Player * player,
            int gridSize )
```

Stawia pudelko na kafelku na ktorym stoi gracz. Funkcja dostepna tylko w trybie debugu.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy |

#### 4.14.1.13 reInitPlayer()

```
void reInitPlayer (
            Player * player,
            int gameTime,
            int gridSize )
```

Reinicjalizuje gracza, tj. czesc danych ustawia mu na nowo

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *gameTime* | czas, w ktorym gracz ma skonczyc plansze |
| *gridSize* | wielkosc mapy |

#### 4.14.1.14 resetPlayerStatics()

```
void resetPlayerStatics ( )
```

Wywoluje funkcje zawierajace zmienne statyczne tak, by je zresetowac

#### 4.14.1.15 shieldSwitch()

```
void shieldSwitch (
            Player * player )
```

Wlacza lub wylacza ulepszenie tarczy gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.14.1.16 waitForGame_over()

```
int waitForGame_over ( )
```

Sprawdza, czy minelo dosc duzo czasu, zeby zakonczyc gre, po przegranej przeciwnika

**Zwraca**

1 kiedy minelo dosc duzo czasu, 0 kiedy nalezy jeszcze czekac

#### 4.14.1.17 waitForNextStageAndSumScore()

```
int waitForNextStageAndSumScore (
            Player * player )
```

Sprawdza, czy minelo dosc duzo czasu, zeby zakonczyc gre, po wygranej przeciwnika oraz przelewa pozostaly czas gracza∗10 do jego punktow.

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

**Zwraca**

1 kiedy minelo dosc duzo czasu, 0 kiedy nalezy jeszcze czekac

## 4.15 Dokumentacja pliku player.h

### Funkcje

- void moveInADirectionAndKickIfPossibleAndCollectAnUpgrade (Direction dir, tile ∗∗bitmap, Player ∗player, int gridSize)
- void initPlayer (Player ∗player, int gameTime, int gridSize)
- void resetPlayerStatics ()
- void reInitPlayer (Player ∗player, int gameTime, int gridSize)
- void openPortal (tile ∗∗bitmap, int gridSize)
- void placeBomb (tile ∗∗bitmap, Player ∗player, int gridSize)
- void placeBox (tile ∗∗bitmap, Player ∗player, int gridSize)
- void damagePlayer (Player ∗player)

- void checkPlayer (tile **bitmap, Player *player, int gameTime, int reset)
- int waitForGame_over ()
- int waitForNextStageAndSumScore (Player *player)
- void increaseBombFireRange (Player *player)
- void decreaseBombFireRange (Player *player)
- void increaseBombsAvalible (Player *player)
- void decreaseBombsAvalible (Player *player)
- void shieldSwitch (Player *player)
- void kickSwitch (Player *player)

### 4.15.1 Dokumentacja funkcji

#### 4.15.1.1 checkPlayer()

```
void checkPlayer (
            tile ** bitmap,
            Player * player,
            int gameTime,
            int reset )
```

Stawia pudelko na kafelku na ktorym stoi gracz. Funkcja dostepna tylko w trybie debugu.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gameTime* | czas, w ktorym gracz ma skonczyc plansze |
| *reset* | jesli 1 to nalezy zresetowac wartosci statycze w funkcji i z niej wyjsc, bez wykonywania reszty jej ciala |

#### 4.15.1.2 damagePlayer()

```
void damagePlayer (
            Player * player )
```

Zabija gracza, jesli ten nie ma aktywnego ulepszenia tarczy. Jesli ma, to po chwili sie je wylacza.

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.3 decreaseBombFireRange()

```
void decreaseBombFireRange (
            Player * player )
```

Obniza zasieg bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.4 decreaseBombsAvalible()

```
void decreaseBombsAvalible (
            Player * player )
```

Zmniejsza ilosc dostepnych bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.5 increaseBombFireRange()

```
void increaseBombFireRange (
            Player * player )
```

Zwieksza zasieg bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.6 increaseBombsAvalible()

```
void increaseBombsAvalible (
            Player * player )
```

Zwieksza ilosc dostepnych bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.7 initPlayer()

```
void initPlayer (
            Player * player,
            int gameTime,
            int gridSize )
```

Inicjalizuje gracza, tj. ustawia mu wszystkie dane

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *gameTime* | czas, w ktorym gracz ma skonczyc plansze |
| *gridSize* | wielkosc mapy |

#### 4.15.1.8 kickSwitch()

```
void kickSwitch (
            Player * player )
```

Wlacza lub wylacza ulepszenie kopania bomb gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.9 moveInADirectionAndKickIfPossibleAndCollectAnUpgrade()

```
void moveInADirectionAndKickIfPossibleAndCollectAnUpgrade (
            Direction dir,
            tile ** bitmap,
            Player * player,
            int gridSize )
```

Przesuwa gracza, jesli moze. Jesli na nowym kafelku bylo ulepszenie, to je zbiera, a jesli na nowym kafelku byla bomba, a gracz ma ulepszenie kopania bomb, to bomba ta zostanie kopnieta

**Parametry**

| | |
|---|---|
| *dir* | kierunek przesuniecia gracza |
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy |

#### 4.15.1.10 openPortal()

```
void openPortal (
            tile ** bitmap,
            int gridSize )
```

Otwiera portal, czyli zamienia srodkowy kafelek mapy na kafelek portalu

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *gridSize* | wielkosc mapy |

#### 4.15.1.11 placeBomb()

```
void placeBomb (
            tile ** bitmap,
            Player * player,
            int gridSize )
```

Stawia bombe na kafelku na ktorym stoi gracz

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy |

#### 4.15.1.12 placeBox()

```
void placeBox (
            tile ** bitmap,
            Player * player,
            int gridSize )
```

Stawia pudelko na kafelku na ktorym stoi gracz. Funkcja dostepna tylko w trybie debugu.

**Parametry**

| | |
|---|---|
| *bitmap* | wskaznik na mape gry |
| *player* | wskaznik na gracza |
| *gridSize* | wielkosc mapy |

#### 4.15.1.13 reInitPlayer()

```
void reInitPlayer (
            Player * player,
            int gameTime,
            int gridSize )
```

Reinicjalizuje gracza, tj. czesc danych ustawia mu na nowo

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |
| *gameTime* | czas, w ktorym gracz ma skonczyc plansze |
| *gridSize* | wielkosc mapy |

#### 4.15.1.14 resetPlayerStatics()

```
void resetPlayerStatics ( )
```

Wywoluje funkcje zawierajace zmienne statyczne tak, by je zresetowac

#### 4.15.1.15 shieldSwitch()

```
void shieldSwitch (
            Player * player )
```

Wlacza lub wylacza ulepszenie tarczy gracza. Dostepne tylko w trybie debugowania

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

#### 4.15.1.16 waitForGame_over()

```
int waitForGame_over ( )
```

Sprawdza, czy minelo dosc duzo czasu, zeby zakonczyc gre, po przegranej przeciwnika

**Zwraca**

1 kiedy minelo dosc duzo czasu, 0 kiedy nalezy jeszcze czekac

#### 4.15.1.17 waitForNextStageAndSumScore()

```
int waitForNextStageAndSumScore (
            Player * player )
```

Sprawdza, czy minelo dosc duzo czasu, zeby zakonczyc gre, po wygranej przeciwnika oraz przelewa pozostaly czas gracza∗10 do jego punktow.

**Parametry**

| | |
|---|---|
| *player* | wskaznik na gracza |

**Zwraca**

1 kiedy minelo dosc duzo czasu, 0 kiedy nalezy jeszcze czekac

## 4.16 Dokumentacja pliku TestArg.h

### Funkcje

- char ∗ check_param (int argc, char ∗∗argv, int ∗howManyEnemies, int ∗howManyBoxes, int ∗howMany↩ Upgrades, int ∗gridSize, int ∗gameTime, int ∗debugMode, int ∗helpSwitch)

### 4.16.1 Dokumentacja funkcji

#### 4.16.1.1 check_param()

```
char* check_param (
            int argc,
            char ** argv,
            int * howManyEnemies,
            int * howManyBoxes,
            int * howManyUpgrades,
            int * gridSize,
            int * gameTime,
            int * debugMode,
            int * helpSwitch )
```

Sprawdza poprawnosc podanych do funkcji main argumentow i zwraca komunikaty ewentualnych bledow.

**Parametry**

| | | |
|---|---|---|
| | *argc* | Ilosc argumentow przekazanych do funkcji main (tj. wraz z wywolaniem programu). |
| | *argv* | Dwuwymiarowa tablica przechowujaca argumenty przekazane do funkcji main (tj. wraz z wywolaniem programu). |
| out | *howManyEnemies* | Ile nalezy naniesc na plansze przeciwnikow. |
| out | *howManyBoxes* | Ile nalezy naniesc na plansze pudelek. |
| out | *howManyUpgrades* | Ile nalezy naniesc na plansze ulepszen. |
| out | *gridSize* | Wielkosc mapy gry. |
| out | *gameTime* | Czas w jakim nalezy wygrac plansze. |
| out | *debugMode* | Czy nalezy wlaczyc gre w trybie debugu (dostepne dodatkowe opcje). 1 - tak, 0 - nie. |
| out | *helpSwitch* | Czy nalezy wyswietlic pomoc. 1 - tak, 0 - nie. |

**Zwraca**

Ciag znakow zawierajacy wykryte w funkcji bledy w argumentach, pomoc lub ciag pusty, jesli zadnych bledow nie bylo.

## 4.17 Dokumentacja pliku TestyArg.c

```
#include "enumsStructsMacros.h"
#include "TestArg.h"
#include <string.h>
#include <stdio.h>
```

### Funkcje

- char * check_param (int argc, char **argv, int *howManyEnemies, int *howManyBoxes, int *howMany↩ Upgrades, int *gridSize, int *gameTime, int *debugMode, int *helpSwitch)

### 4.17.1 Dokumentacja funkcji

#### 4.17.1.1 check_param()

```
char* check_param (
            int argc,
            char ** argv,
            int * howManyEnemies,
            int * howManyBoxes,
            int * howManyUpgrades,
            int * gridSize,
            int * gameTime,
            int * debugMode,
            int * helpSwitch )
```

Sprawdza poprawnosc podanych do funkcji main argumentow i zwraca komunikaty ewentualnych bledow.

**Parametry**

| | | |
|---|---|---|
| | *argc* | Ilosc argumentow przekazanych do funkcji main (tj. wraz z wywolaniem programu). |
| | *argv* | Dwuwymiarowa tablica przechowujaca argumenty przekazane do funkcji main (tj. wraz z wywolaniem programu). |
| out | *howManyEnemies* | Ile nalezy naniesc na plansze przeciwnikow. |
| out | *howManyBoxes* | Ile nalezy naniesc na plansze pudelek. |
| out | *howManyUpgrades* | Ile nalezy naniesc na plansze ulepszen. |
| out | *gridSize* | Wielkosc mapy gry. |
| out | *gameTime* | Czas w jakim nalezy wygrac plansze. |
| out | *debugMode* | Czy nalezy wlaczyc gre w trybie debugu (dostepne dodatkowe opcje). 1 - tak, 0 - nie. |
| out | *helpSwitch* | Czy nalezy wyswietlic pomoc. 1 - tak, 0 - nie. |

**Zwraca**

Ciag znakow zawierajacy wykryte w funkcji bledy w argumentach, pomoc lub ciag pusty, jesli zadnych bledow nie bylo.

# Skorowidz