

Boombberman

Wygenerowano przez Doxygen 1.8.14

Spis treści

1	Indeks struktur danych	1
1.1	Struktury danych	1
2	Indeks plików	3
2.1	Lista plików	3
3	Dokumentacja struktur danych	5
3.1	Dokumentacja struktury Bomb	5
3.1.1	Opis szczegółowy	5
3.1.2	Dokumentacja pól	5
3.1.2.1	coords	5
3.1.2.2	explosionTimer	6
3.1.2.3	fireCross	6
3.1.2.4	kickDirection	6
3.1.2.5	kickTimer	6
3.1.2.6	size	6
3.1.2.7	statFlag	6
3.2	Dokumentacja struktury Enemy	6
3.2.1	Opis szczegółowy	7
3.2.2	Dokumentacja pól	7
3.2.2.1	coords	7
3.2.2.2	direction	7
3.2.2.3	rottingTime	7
3.2.2.4	status	7

3.2.2.5	type	7
3.3	Dokumentacja struktury Player	8
3.3.1	Opis szczegółowy	8
3.3.2	Dokumentacja pól	8
3.3.2.1	bombList	8
3.3.2.2	brokenShieldTimer	8
3.3.2.3	coords	8
3.3.2.4	newestBomb	9
3.3.2.5	oldestBomb	9
3.3.2.6	score	9
3.3.2.7	status	9
3.3.2.8	timeLeft	9
3.3.2.9	upgrades	9
3.4	Dokumentacja struktury tile	9
3.4.1	Dokumentacja pól	10
3.4.1.1	fireLayers	10
3.4.1.2	type	10
3.4.1.3	upgrade	10
3.5	Dokumentacja struktury XY	10
3.5.1	Dokumentacja pól	10
3.5.1.1	x	10
3.5.1.2	y	10

4 Dokumentacja plików	11
4.1 Dokumentacja pliku draw.c	11
4.1.1 Dokumentacja funkcji	11
4.1.1.1 drawAllEnemies()	12
4.1.1.2 drawBomb()	12
4.1.1.3 drawBombs()	12
4.1.1.4 drawEnemy()	13
4.1.1.5 drawEnemyBlind()	13
4.1.1.6 drawEnemychargerCalm()	14
4.1.1.7 drawEnemychargerUpset()	14
4.1.1.8 drawEnemyRandom()	15
4.1.1.9 drawGrid()	15
4.1.1.10 drawGridBase()	15
4.1.1.11 drawPauseScreen()	16
4.1.1.12 drawPlayer()	16
4.1.1.13 drawPortal()	16
4.1.1.14 drawScoreboard()	16
4.1.1.15 drawSquare()	17
4.1.1.16 drawUpgrade()	17
4.1.1.17 initGrid()	18
4.1.1.18 resetDrawStatics()	18
4.1.1.19 setGILists()	18
4.1.1.20 unSetGIList()	18
4.2 Dokumentacja pliku draw.h	19
4.2.1 Dokumentacja funkcji	19
4.2.1.1 drawAllEnemies()	19
4.2.1.2 drawBomb()	20
4.2.1.3 drawBombs()	20
4.2.1.4 drawEnemy()	20
4.2.1.5 drawEnemyBlind()	21

4.2.1.6	drawEnemychargerCalm()	21
4.2.1.7	drawEnemychargerUpset()	22
4.2.1.8	drawEnemyRandom()	22
4.2.1.9	drawGrid()	23
4.2.1.10	drawGridBase()	23
4.2.1.11	drawPauseScreen()	23
4.2.1.12	drawPlayer()	24
4.2.1.13	drawPortal()	24
4.2.1.14	drawScoreboard()	24
4.2.1.15	drawSquare()	24
4.2.1.16	drawUpgrade()	25
4.2.1.17	initGrid()	25
4.2.1.18	resetDrawStatics()	26
4.2.1.19	setGILists()	26
4.2.1.20	unSetGIList()	26
4.3	Dokumentacja pliku Enemies.c	27
4.3.1	Dokumentacja funkcji	27
4.3.1.1	addEnemy()	27
4.3.1.2	dropEnemies()	28
4.3.1.3	initEnemies()	28
4.3.1.4	manageEnemiesAndScore()	28
4.3.1.5	moveBlind()	29
4.3.1.6	moveCharger()	29
4.3.1.7	moveEnemy()	30
4.3.1.8	moveRandom()	30
4.3.1.9	pointTowardsPlayerIfPossible()	30
4.3.1.10	resetEnemyStatics()	31
4.4	Dokumentacja pliku Enemies.h	31
4.4.1	Dokumentacja funkcji	31
4.4.1.1	addEnemy()	31

4.4.1.2	dropEnemies()	32
4.4.1.3	initEnemies()	32
4.4.1.4	manageEnemiesAndScore()	33
4.4.1.5	moveBlind()	33
4.4.1.6	moveCharger()	33
4.4.1.7	moveEnemy()	34
4.4.1.8	moveRandom()	34
4.4.1.9	pointTowardsPlayerIfPossible()	35
4.4.1.10	resetEnemyStatics()	35
4.5	Dokumentacja pliku enumsStructsMacros.h	35
4.5.1	Dokumentacja typów wyliczanych	37
4.5.1.1	bombState	37
4.5.1.2	Direction	37
4.5.1.3	EnemyType	38
4.5.1.4	PauseScreenType	38
4.5.1.5	Status	38
4.5.1.6	tileType	39
4.5.1.7	upgrades	39
4.6	Dokumentacja pliku fonts.h	41
4.6.1	Dokumentacja definicji typów	41
4.6.1.1	sign	41
4.6.2	Dokumentacja funkcji	41
4.6.2.1	translateWord()	41
4.6.2.2	writeDynamicData()	42
4.6.2.3	writePauseMessage()	42
4.6.2.4	writePauseMessageScore()	42
4.6.2.5	writeStaticData()	43
4.7	Dokumentacja pliku highscoreFile.c	43
4.7.1	Dokumentacja funkcji	43
4.7.1.1	getHighscore()	43

4.7.1.2	setHighscore()	43
4.8	Dokumentacja pliku highscoreFile.h	44
4.8.1	Dokumentacja funkcji	44
4.8.1.1	getHighscore()	44
4.8.1.2	setHighscore()	44
4.9	Dokumentacja pliku main.c	45
4.9.1	Dokumentacja funkcji	45
4.9.1.1	error_callback()	45
4.9.1.2	init()	46
4.9.1.3	key_callback()	46
4.9.1.4	pauseGame()	47
4.9.1.5	resetTimes()	47
4.9.1.6	setDefaultsIfNecessary()	48
4.9.1.7	unPauseGame()	48
4.10	Dokumentacja pliku manage bombs.C	48
4.10.1	Dokumentacja funkcji	49
4.10.1.1	cleanTile()	49
4.10.1.2	cleanUp()	49
4.10.1.3	explodeBombAddPoints()	50
4.10.1.4	getBombID()	50
4.10.1.5	moveBombData()	50
4.10.1.6	moveCheckExplodeAndCleanUpBombs()	51
4.10.1.7	moveTile()	51
4.10.1.8	spreadFire()	51
4.11	Dokumentacja pliku manage bombs.h	52
4.11.1	Dokumentacja funkcji	52
4.11.1.1	cleanTile()	52
4.11.1.2	cleanUp()	52
4.11.1.3	explodeBombAddPoints()	54
4.11.1.4	getBombID()	54

4.11.1.5	moveBombData()	55
4.11.1.6	moveCheckExplodeAndCleanUpBombs()	55
4.11.1.7	moveTile()	55
4.11.1.8	spreadFire()	56
4.12	Dokumentacja pliku mapMemAndGen.c	56
4.12.1	Dokumentacja funkcji	56
4.12.1.1	allocateMap()	56
4.12.1.2	deallocateMap()	57
4.12.1.3	generateMap()	57
4.13	Dokumentacja pliku mapMemAndGen.h	57
4.13.1	Dokumentacja funkcji	57
4.13.1.1	allocateMap()	58
4.13.1.2	deallocateMap()	58
4.13.1.3	generateMap()	58
4.14	Dokumentacja pliku player.c	59
4.14.1	Dokumentacja funkcji	59
4.14.1.1	checkPlayer()	59
4.14.1.2	damagePlayer()	60
4.14.1.3	decreaseBombFireRange()	60
4.14.1.4	decreaseBombsAvalible()	60
4.14.1.5	increaseBombFireRange()	60
4.14.1.6	increaseBombsAvalible()	61
4.14.1.7	initPlayer()	61
4.14.1.8	kickSwitch()	61
4.14.1.9	moveInADirectionAndKickIfPossibleAndCollectAnUpgrade()	62
4.14.1.10	openPortal()	62
4.14.1.11	placeBomb()	62
4.14.1.12	placeBox()	63
4.14.1.13	reInitPlayer()	63
4.14.1.14	resetPlayerStatics()	63

4.14.1.15 shieldSwitch()	63
4.14.1.16 waitForGame_over()	64
4.14.1.17 waitForNextStageAndSumScore()	64
4.15 Dokumentacja pliku player.h	64
4.15.1 Dokumentacja funkcji	65
4.15.1.1 checkPlayer()	65
4.15.1.2 damagePlayer()	65
4.15.1.3 decreaseBombFireRange()	66
4.15.1.4 decreaseBombsAvalible()	66
4.15.1.5 increaseBombFireRange()	66
4.15.1.6 increaseBombsAvalible()	66
4.15.1.7 initPlayer()	67
4.15.1.8 kickSwitch()	67
4.15.1.9 moveInADirectionAndKickIfPossibleAndCollectAnUpgrade()	67
4.15.1.10 openPortal()	68
4.15.1.11 placeBomb()	68
4.15.1.12 placeBox()	68
4.15.1.13 reInitPlayer()	70
4.15.1.14 resetPlayerStatics()	70
4.15.1.15 shieldSwitch()	70
4.15.1.16 waitForGame_over()	70
4.15.1.17 waitForNextStageAndSumScore()	71
4.16 Dokumentacja pliku TestArg.h	71
4.16.1 Dokumentacja funkcji	71
4.16.1.1 check_param()	71
4.17 Dokumentacja pliku TestyArg.c	72
4.17.1 Dokumentacja funkcji	72
4.17.1.1 check_param()	72

Rozdział 1

Indeks struktur danych

1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Bomb	5
Enemy	6
Player	8
tile	9
XY	10

Rozdział 2

Indeks plików

2.1 Lista plików

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

draw.c	11
draw.h	19
Enemies.c	27
Enemies.h	31
enumsStructsMacros.h	35
fonts.h	41
highscoreFile.c	43
highscoreFile.h	44
main.c	45
manage bombs.C	48
manage bombs.h	52
mapMemAndGen.c	56
mapMemAndGen.h	57
player.c	59
player.h	64
TestArg.h	71
TestyArg.c	72

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 względem 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 zostala 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 względem mapy.

3.2.2.2 direction

[Direction](#) Enemy::direction

Kierunek poruszania się przeciwnika.

3.2.2.3 rottingTime

float Enemy::rottingTime

Jak długo należy rysować zwłoki przeciwnika po jego śmierci.

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 zawierająca bomby postawione przez gracza.

3.3.2.2 brokenShieldTimer

```
double Player::brokenShieldTimer
```

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

3.3.2.3 coords

```
XY Player::coords
```

Koordynaty gracza względem mapy.

3.3.2.4 newestBomb

```
int Player::newestBomb
```

Indeks najpóźniej położonej i jeszcze aktywnej bomby w tablicy bomb.

3.3.2.5 oldestBomb

```
int Player::oldestBomb
```

Indeks najwcześniej położonej 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 którym gracz zacznie tracić punkty, aż się wyzeruje, 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
- [fireLayers](#) int
- [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
```

Koordinat x.

3.5.1.2 y

```
int XY::y
```

Koordinat 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 `setGILists` (`int listArray[6]`, `int gridSize`)
- void `unSetGIList` (`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 którym wypisuje punkty gracza, najwyższe dotychczasowe punkty oraz pozostały czas gry

Parametry

<i>enemyArray</i>	tablica przechowująca dane wszystkich przeciwników
<i>gridSize</i>	wielkość mapy (NxN)
<i>howMany</i>	ile przeciwników jest do narysowania

4.1.1.2 drawBomb()

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

Rysuje pojedynczą bombę o określonym określonym rozmiarze w określonym miejscu

Parametry

<i>x</i>	koordynat szerokości względem okna
<i>y</i>	koordynat wysokości względem okna
<i>sizeX</i>	szerokość obiektu względem okna
<i>sizeY</i>	wysokość obiektu względem okna
<i>bomb</i>	wskaznik na bombę do narysowania

4.1.1.3 drawBombs()

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

Wywołuje rysowanie wszystkich bomb w określonych miejscach

Parametry

<i>player</i>	wskaznik na gracza (gracz przechowuje bomby)
<i>gridSize</i>	wielkość mapy (NxN)

4.1.1.4 drawEnemy()

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

Wywołuje funkcje rysowania przeciwnika względem jego typu w określonym miejscu

Parametry

<i>x</i>	Koordinat szerokosci względem okna.
<i>y</i>	Koordinat wysokosci względem okna.
<i>sizeX</i>	Szerokosc obiektu względem okna.
<i>sizeY</i>	Wysokosc obiektu względem okna.
<i>enemy</i>	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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci względem okna
<i>y</i>	koordynat wysokosci względem okna
<i>sizeX</i>	szerokosc obiektu względem okna
<i>sizeY</i>	wysokosc obiektu względem 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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokości względem okna
<i>y</i>	koordynat wysokości względem okna
<i>sizeX</i>	szerokość obiektu względem okna
<i>sizeY</i>	wysokość obiektu względem 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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokości względem okna
<i>y</i>	koordynat wysokości względem okna
<i>sizeX</i>	szerokość obiektu względem okna
<i>sizeY</i>	wysokość obiektu względem 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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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

<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	wielkosc mapy (NxN)

4.1.1.10 drawGridBase()

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

Funkcja rysujaca podstawe mapy - tlo, sciezki, kolumny

Parametry

<i>gridSize</i>	wielkosc mapy (NxN)
-----------------	---------------------

4.1.1.11 drawPauseScreen()

```
void drawPauseScreen ( )
```

Rysuje tło ekran pauzy

4.1.1.12 drawPlayer()

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

Rysuje gracza w określonym miejscu

Parametry

<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkość mapy (NxN)

4.1.1.13 drawPortal()

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

Rysuje portal o określonym rozmiarze i w określonym miejscu

Parametry

<i>x</i>	koordynat szerokości względem okna
<i>y</i>	koordynat wysokości względem okna
<i>sizeX</i>	szerokość obiektu względem okna
<i>sizeY</i>	wysokość obiektu względem okna

4.1.1.14 drawScoreboard()

```
void drawScoreboard ( )
```

Rysuje tło panel obok mapy, na którym wypisane będą punkty gracza, najwyższe dotychczasowe punkty oraz pozostały 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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	wysokosc obiektu wzgledem okna

4.1.1.17 initGrid()

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

Inicjalizuje rzeczy wymagające inicjalizacji z tego pliku

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>gridSize</i>	wielkość mapy (NxN)
<i>howManyBoxes</i>	ile na mapie znajduje się pudeł
<i>howManyUpgrades</i>	ile na mapie znajduje się ulepszeń

4.1.1.18 resetDrawStatics()

```
void resetDrawStatics ( )
```

Wywołuje funkcje zawierające zmienne statyczne tak, by je zresetować

4.1.1.19 setGILists()

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

Ustawia gIListy dla optymalniejszego rysowania

Parametry

<i>listArray</i>	Tablica przechowująca indeksy gIList.
<i>gridSize</i>	wielkość mapy

4.1.1.20 unSetGIList()

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

Usuwa stworzone wcześniej gIListy

Parametry

<i>listArray</i>	Tablica przechowująca 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 którym wypisuje punkty gracza, najwyższe dotychczasowe punkty oraz pozostały czas gry

Parametry

<i>enemyArray</i>	tablica przechowująca dane wszystkich przeciwników
<i>gridSize</i>	wielkość mapy (NxN)
<i>howMany</i>	ile przeciwników 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

<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	wysokosc obiektu wzgledem okna
<i>bomb</i>	wskaznik na bombe do narysowania

4.2.1.3 drawBombs()

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

Wywołuje rysowanie wszystkich bomb w okreslonych miejscach

Parametry

<i>player</i>	wskaznik na gracza (gracz przechowuje bomby)
<i>gridSize</i>	wielkosc mapy (NxN)

4.2.1.4 drawEnemy()

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

Wywołuje funkcje rysowania przeciwnika wzgledem jego typu w okreslonym miejscu

Parametry

<i>x</i>	Koordinat szerokosci wzgledem okna.
<i>y</i>	Koordinat wysokosci wzgledem okna.
<i>sizeX</i>	Szerokosc obiektu wzgledem okna.
<i>sizeY</i>	Wysokosc obiektu wzgledem okna.
<i>enemy</i>	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

<i>R</i>	Skladowa koloru RGB
<i>G</i>	Skladowa koloru RGB
<i>B</i>	Skladowa koloru RGB
<i>x</i>	koordinat szerokosci wzgledem okna
<i>y</i>	koordinat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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

<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	wielkosc mapy (NxN)

4.2.1.10 drawGridBase()

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

Funkcja rysujaca podstawe mapy - tlo, sciezki, kolumny

Parametry

<i>gridSize</i>	wielkosc mapy (NxN)
-----------------	---------------------

4.2.1.11 drawPauseScreen()

```
void drawPauseScreen ( )
```

Rysuje tło ekran pauzy

4.2.1.12 drawPlayer()

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

Rysuje gracza w określonym miejscu

Parametry

<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkosc mapy (NxN)

4.2.1.13 drawPortal()

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

Rysuje portal o określonym rozmiarze i w określonym miejscu

Parametry

<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	wysokosc obiektu wzgledem okna

4.2.1.14 drawScoreboard()

```
void drawScoreboard ( )
```

Rysuje tło panel obok mapy, na którym wypisane będą punkty gracza, najwyższe dotychczasowe punkty oraz pozostały 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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	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 określonym kolorze, określonym rozmiarze i w określonym miejscu

Parametry

<i>R</i>	Składowa koloru RGB
<i>G</i>	Składowa koloru RGB
<i>B</i>	Składowa koloru RGB
<i>x</i>	koordynat szerokosci wzgledem okna
<i>y</i>	koordynat wysokosci wzgledem okna
<i>sizeX</i>	szerokosc obiektu wzgledem okna
<i>sizeY</i>	wysokosc obiektu wzgledem okna

4.2.1.17 initGrid()

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

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

Inicjalizuje rzeczy wymagające inicjalizacji z tego pliku

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>gridSize</i>	wielkość mapy (NxN)
<i>howManyBoxes</i>	ile na mapie znajduje się pudeł
<i>howManyUpgrades</i>	ile na mapie znajduje się ulepszeń

4.2.1.18 resetDrawStatics()

```
void resetDrawStatics ( )
```

Wywołuje funkcje zawierające zmienne statyczne tak, by je zresetować

4.2.1.19 setGILists()

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

Ustawia gIListy dla optymalniejszego rysowania

Parametry

<i>listArray</i>	Tablica przechowująca indeksy gIList.
<i>gridSize</i>	wielkość mapy

4.2.1.20 unSetGIList()

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

Usuwa stworzone wcześniej gIListy

Parametry

<i>listArray</i>	Tablica przechowująca indeksy gIList.
------------------	---------------------------------------

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

<i>enemyArray</i>	Tablica przechowująca dane wszystkich przeciwników.
<i>ID</i>	Indeks nowego przeciwnika w tablicy przeciwników.
<i>type</i>	Typ przeciwnika.
<i>coords</i>	Koordinaty nowego przeciwnika względem mapy.
<i>dir</i>	Kierunek poruszania się 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

<i>enemyArray</i>	Tablica przechowujaca dane wszystkich przeciwnikow.
<i>bitmap</i>	Wskaźnik na mape gry.
<i>howMany</i>	Ile przeciwnikow jest do wygenerowania.
<i>gridSize</i>	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

<i>enemyArray</i>	Tablica przechowujaca dane wszystkich przeciwnikow.
<i>bitmap</i>	Wskaźnik na mape gry.
<i>howMany</i>	Ile przeciwnikow jest do wygenerowania.
<i>gridSize</i>	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 )
```

Obsługuje 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

<i>enemyArray</i>	tablica przechowująca dane wszystkich przeciwników.
<i>player</i>	Wskaźnik na gracza.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>howMany</i>	Ile jest przeciwników.
<i>gridSize</i>	Wielkość mapy (NxN).
<i>reset</i>	Jeśli 1 to należy zresetować wartości statyczne w funkcji i z niej wyjść, bez wykonywania reszty jej ciała.

Zwraca

Liczba 0 kiedy żyje przynajmniej jeden przeciwnik, liczba 1 jeśli wszyscy są 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

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość 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

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>player</i>	Wskaźnik na gracza.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość 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

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>rush</i>	Przełącznik, czy przeciwnik ma się przebijać przez pudelka.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość mapy (NxN).

Zwraca

Liczba 0 jeśli się nie udało przesunąć przeciwnika, liczba 1 w przypadku kiedy się udało, 2 jeśli kierunek w którym próbowano się poruszyć był inny przeciwnik.

4.3.1.8 moveRandom()

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

Przesuwa przeciwnika typu 'random'.

Parametry

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość 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

<i>enemy</i>	Wskaźnik na przeciwnika, ktorego sprawdzane jest pole widzenia.
<i>player</i>	Wskaźnik 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 ( )
```

Wywołuje 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 gridSize)

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

<i>enemyArray</i>	Tablica przechowująca dane wszystkich przeciwników.
<i>ID</i>	Indeks nowego przeciwnika w tablicy przeciwników.
<i>type</i>	Typ przeciwnika.
<i>coords</i>	Koordynaty nowego przeciwnika względem mapy.
<i>dir</i>	Kierunek poruszania się nowego przeciwnika.

4.4.1.2 dropEnemies()

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

Stawia nowych przeciwników na planszy.

Parametry

<i>enemyArray</i>	Tablica przechowująca dane wszystkich przeciwników.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>howMany</i>	Ile przeciwników jest do wygenerowania.
<i>gridSize</i>	Wielkość mapy (NxN).

4.4.1.3 initEnemies()

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

Inicjalizuje rzeczy wymagające inicjalizacji z tego pliku.

Parametry

<i>enemyArray</i>	Tablica przechowująca dane wszystkich przeciwników.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>howMany</i>	Ile przeciwników jest do wygenerowania.
<i>gridSize</i>	Wielkość 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 )
```

Obsługuje przeciwników, tj. przesuwa ich, sprawdza czy nie zostali zabici przez gracza (stoja na ogniu), dodaje punkty graczowi, jeśli przeciwnik umiera i sprawdza, czy ktoreś z przeciwników jeszcze żyje.

Parametry

<i>enemyArray</i>	tablica przechowująca dane wszystkich przeciwników.
<i>player</i>	Wskaźnik na gracza.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>howMany</i>	Ile jest przeciwników.
<i>gridSize</i>	Wielkość mapy (NxN).
<i>reset</i>	Jeśli 1 to należy zresetować wartości statyczne w funkcji i z niej wyjść, bez wykonywania reszty jej ciała.

Zwraca

Liczba 0 kiedy żyje przynajmniej jeden przeciwnik, liczba 1 jeśli wszyscy są 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

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość 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

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>player</i>	Wskaźnik na gracza.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość mapy (NxN).

4.4.1.7 moveEnemy()

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

Probuje przesunąć przeciwnika w jego kierunku o jeden kafelek.

Parametry

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>rush</i>	Przełącznik, czy przeciwnik ma się przebijac przez pudełko.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość mapy (NxN).

Zwraca

Liczba 0 jeśli się nie udało przesunąć przeciwnika, liczba 1 w przypadku kiedy się udało, 2 jeśli kierunek w którym próbowano się poruszyć był inny przeciwnik.

4.4.1.8 moveRandom()

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

Przesuwa przeciwnika typu 'random'.

Parametry

<i>enemy</i>	Wskaźnik na przeciwnika do przesunięcia.
<i>bitmap</i>	Wskaźnik na mapę gry.
<i>gridSize</i>	Wielkość mapy (NxN).

4.4.1.9 pointTowardsPlayerIfPossible()

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

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

Parametry

<i>enemy</i>	Wskaźnik na przeciwnika, którego sprawdzane jest pole widzenia.
<i>player</i>	Wskaźnik na gracza.

Zwraca

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

4.4.1.10 resetEnemyStatics()

```
void resetEnemyStatics ( )
```

Wywołuje funkcje zawierające zmienne statyczne tak, by je zresetować.

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_AVAILABLE 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_AVAILABLE 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

<code>empty</code>	Bomba nie jest polozona.
<code>placed</code>	Bomba jest polozona i jeszcze nie wybuchla.
<code>exploding</code>	Bomba wlasnie wybuchnela.
<code>exploded</code>	Bomba jest po wybuchu.
<code>toBeCleaned</code>	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 żywego gracza i przeciwnika.
winning	Status gracza jeśli wejdzie do portalu.
dying	Status przeciwnika, który przed chwilą umarł i jeszcze nie minelo dość dużo czasu, żeby wyłączyć rysowanie jego korpusu.
rushing	Status przeciwnika typu 'charger', jeśli 'zobaczył' (sposób 'widzenia' opisany w komentarzu w do enumeratora EnrmyType) gracza.
calm	Status przeciwnika typu 'charger' jeśli nie 'widział' (sposób 'widzenia' opisany w komentarzu w do enumeratora EnrmyType) przez jakiś czas gracza.
appearing	Status gracza pojawiającego się na mapie (pierwszy moment gry).

4.5.1.6 tileType

```
enum tileType
```

Typy kafelek mapy

Wartości wyliczeń

pathTile	Kafelek ścieżki - gracz, przeciwnik oraz przesuwająca się bomba mogą się po nim poruszać.
boxTile	Kafelek pudełka - wybuch bomby może go zniszczyć, poruszająca bomba się przed nim zatrzyma, nic nie może przez niego przejść, poza przeciwnikiem typu 'charger' w jego drugiej fazie.
pillarTile	Kafelek kolumny - nic nie może przez niego przejść, nie można go w żaden sposób zniszczyć, przeciwnik typu 'charger' nie zobaczy przeciwnika, jeśli na jego polu widzenia znajduje się ten kafelek.
bombTile	Kafelek na którym znajduje się bomba - nic przez niego nie może przejść, poza graczem, który ma ulepszenie 'bombKick', w takim wypadku gracz może wejść na ten kafelek, a bomba na nim się znajdująca zacznie się przesuwać.
fireTile	Kafelek na którym znajduje się ogień - wszystko może na niego wejść, przy czym przeciwnicy i gracz bez ulepszenia 'shield' znajdujący się na tym kafelku umierają. Gracz posiadający ulepszenie shield straci je.
firedBox	Kafelek na którym znajduje się pudełko, które dosięgnęła eksplozja bomby - przeciwnicy nie mogą przez niego przejść poza przeciwnikiem typu 'charger' w jego drugiej fazie, który wchodzić na niego umrze. Dla gracza zachowuje się jak kafelek 'fireTile'.
portalTile	Kafelek, który pojawia się na środku mapy w momencie, kiedy wszyscy przeciwnicy zostaną pokonani. Kiedy gracz na niego wejdzie, to rozpocznie podsumowanie punktów i pauza typu zwycięstwo.
enemyTile	Kafelek na którym znajduje się przeciwnik - inni przeciwnicy nie mogą na niego wejść, gracz wchodzić na nie otrzymuje obrażenia. Nie można na nim postawić bomby, a poruszająca bomba się przed nim zatrzyma.

4.5.1.7 upgrades

```
enum upgrades
```

Typy ulepszen

Wartości wyliczeń

fireRange	Zasięg bomby.
bombsAvailable	Ile bomb na raz może być postawionych na mapie.
bombKick	Czy gracz może "kopnąć" bombę, tj. sprawić, żeby bomba przesuwała się w określonym kierunku.
shield	Czy gracz ma tarczę. Jeśli tak, to może stanąć na ogniu albo spotkać przeciwnika bez natychmiastowej śmierci, ale wtedy tarcza w ciągu 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 reprezentująca znak w postaci bitów - pixeli

4.6.2 Dokumentacja funkcji

4.6.2.1 translateWord()

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

Zamienia wejściowy ciąg znaków, tą tablice wskaźników na tablice reprezentujące znaki w postaci bitów - pixeli

Parametry

out	<i>signs</i>	wskaznik na wskaznik na tablice wskaznikow na tablice reprezentujace znaki w postaci bitow - pixeli
	<i>word</i>	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

<i>highscore</i>	najwyzsza dotychczasowa ilosc punktow
<i>score</i>	aktualna ilosc punktow
<i>timeLeft</i>	pozostaly czas gracza

4.6.2.3 writePauseMessage()

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

Wypisuje na okno pauzy wiadomosc, zalezaca od typu pauzy

Parametry

<i>pauseType</i>	typ pauzy
------------------	-----------

4.6.2.4 writePauseMessageScore()

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

Wypisuje na okno pauzy aktualna ilosc punktow

Parametry

<code>score</code>	aktualna ilosc punktow
--------------------	------------------------

4.6.2.5 `writeStaticData()`

```
void writeStaticData ( )
```

Zapisuje w obszarze okna obok mapy tytuły informacji: aktualne punkty, największa 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 największej ilosci punktow

Parametry

	<i>in</i>	wskaznik na strukture pliku
<code>out</code>	<i>pobrana</i>	z pliku wartosc dotychczasowej największej ilosci punktow

4.7.1.2 `setHighscore()`

```
void setHighscore (
```

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

Zapisuje do pliku wartosc dotychczasowej najwiekszej ilosci punktow

Parametry

<i>out</i>	wskaznik na strukture pliku
<i>score</i>	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

	<i>in</i>	wskaznik na strukture pliku
<i>out</i>	<i>pobrana</i>	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

<i>out</i>	wskaznik na strukture pliku
<i>score</i>	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` ***bitmapSetter, `Player` *playerSetter, int *debugModeSetter, int *gridSizeSetter)
- void `error_callback` (int error, const char *description)
- void `setDefaultslfNecessary` (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 *gridSize, 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

<i>error</i>	numer bledu
<i>description</i>	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

<i>enemy</i>	wskaznik na przeciwnika do przesunięcia
<i>bitmap</i>	wskaznik na wskaznik na mape gry
<i>pauseType</i>	Wskaznik na typ pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji.
<i>dTime</i>	Wskaznik na zmienna pamietajaca czas gry sprzed pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji.
<i>player</i>	wskaznik na gracza
<i>howManyEnemies</i>	Ile ma sie znajdowac przeciwnikow na mapie.
<i>howManyBoxes</i>	Ile ma sie znajdowac pudelek na mapie.
<i>howManyUpgrades</i>	Ile ma sie znajdowac ulepszen na mapie.
<i>gridSize</i>	wielkosc mapy
<i>gameTime</i>	czas gry
<i>debugMode</i>	zmienna ustawiajaca debugMode gry
<i>[out]</i>	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]
```

Wywołuje odpowiednie funkcje odpowiadające wciśnięciu odpowiednich klawiszy

Parametry

<i>window</i>	wskaznik na strukture okna glfw.
<i>key</i>	Jaki klawisz.
<i>scancode</i>	
<i>action</i>	Akcja klawisza (naciśnięty czy puszczony).
<i>mods</i>	
<i>bitmap</i>	wskaznik na wskaznik na mapę gry. Służy do ustawienia zmiennych statycznych wewnątrz funkcji)
<i>playerSetter</i>	wskaznik gracza. Służy do ustawienia zmiennych statycznych wewnątrz funkcji)
<i>debugModeSetter</i>	wskaznik zmienna ustawiająca debugMode gry. Służy do ustawienia zmiennych statycznych wewnątrz funkcji)
<i>gridSizeSetter</i>	wskaznik zmienna przechowująca wielkość mapy. Służy do ustawienia zmiennych statycznych wewnątrz funkcji)

4.9.1.4 pauseGame()

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

Pauzuje grę i zapisuje aktualny czas

Parametry

<i>pauseSetter</i>	Wskaznik na typ pauzy. Służy do ustawienia zmiennej statycznej wewnątrz funkcji.
<i>dTimeSetter</i>	Wskaznik na zmienną pamiętającą czas gry przed pauzą. Służy do ustawienia zmiennej statycznej wewnątrz funkcji.

4.9.1.5 resetTimes()

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

Wywołuje funkcje resetujące wszystkie funkcje statyczne oraz resetuje czas

Parametry

out	<i>dTime</i>	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	<i>howManyEnemies</i>	Ile ma sie znajdowac przeciwnikow na mapie.
out	<i>howManyBoxes</i>	Ile ma sie znajdowac pudelek na mapie.
out	<i>howManyUpgrades</i>	Ile ma sie znajdowac ulepszen na mapie.
out	<i>gridSize</i>	wielkosc mapy
out	<i>gameTime</i>	czas gry
out	<i>debugMode</i>	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

<i>pauseSetter</i>	Wskaznik na typ pauzy. Sluzy do ustawienia zmiennej statycznej wewnatrz funkcji.
<i>dTimeSetter</i>	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

<i>currTile</i>	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

<i>currentBombID</i>	indeks w tablicy bomb gracza aktualnej bomby
<i>player</i>	wskaznik na gracza
<i>bitmap</i>	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 ogień po mapie zgodnie z zasięgiem danej bomby zwiększa ilość wolnych bomb gracza

Parametry

<i>bomb</i>	wskaznik na bombe, która należy eksplodować
<i>player</i>	wskaznik na gracza
<i>bitmap</i>	wskaznik na mapę gry
<i>gridSize</i>	wielkość mapy

4.10.1.4 getBombID()

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

Znajduje ID w tablicy bomb gracza bomby o określonych współrzędnych

Parametry

<i>player</i>	wskaznik na gracza
<i>x</i>	koordynat szerokości względem mapy
<i>y</i>	koordynat wysokości względem mapy

Zwraca

indeks, pod którym w tablicy bomb gracza znajduje się bomba o podanych współrzędnych

4.10.1.5 moveBombData()

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

Przekazuje dane bomby A do bomby B.

Parametry

<i>A</i>	bomba, ktorej dane sa przekazywane
<i>B</i>	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

<i>player</i>	wskaznik na gracza
<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	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

<i>dir</i>	kierunek poruszania sie bomby
<i>currBomb</i>	wskaznik na aktualnie przesuwana bombe
<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	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

	<i>currTile</i>	wskaznik na aktualny kafelek do sprawdzenia
out	<i>breaker</i>	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

<i>currTile</i>	Kafelek do obsluzenia.
-----------------	------------------------

4.11.1.2 [cleanUp\(\)](#)

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

Usuwa ogień pozostały po eksplozji bomby, jeśli minęło dość dużo czasu od jej wybuchu

Parametry

<i>currentBombID</i>	indeks w tablicy bomb gracza aktualnej bomby
<i>player</i>	wskaznik na gracza
<i>bitmap</i>	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 ogień po mapie zgodnie z zasięgiem danej bomby zwiększa ilość wolnych bomb gracza

Parametry

<i>bomb</i>	wskaznik na bombe, która należy eksplodować
<i>player</i>	wskaznik na gracza
<i>bitmap</i>	wskaznik na mapę gry
<i>gridSize</i>	wielkość mapy

4.11.1.4 getBombID()

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

Znajduje ID w tablicy bomb gracza bomby o określonych współrzędnych

Parametry

<i>player</i>	wskaznik na gracza
<i>x</i>	koordynat szerokości względem mapy
<i>y</i>	koordynat wysokości względem mapy

Zwraca

indeks, pod którym w tablicy bomb gracza znajduje się bomba o podanych współrzędnych

4.11.1.5 moveBombData()

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

Przekazuje dane bomby A do bomby B.

Parametry

<i>A</i>	bomba, ktorej dane sa przekazywane
<i>B</i>	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

<i>player</i>	wskaznik na gracza
<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	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

<i>dir</i>	kierunek poruszania sie bomby
<i>currBomb</i>	wskaznik na aktualnie przesuwana bombe
<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	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

	<i>currTile</i>	wskaznik na aktualny kafelek do sprawdzenia
out	<i>breaker</i>	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 [deallocMap](#) (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

<i>bitmap</i>	wskaznik na wskaznik na mape gry
<i>gridSize</i>	wielkosc mapy

4.12.1.2 deallocateMap()

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

Dealokuje pamiec mapy o okreslonej wielkosci.

Parametry

<i>bitmap</i>	wskaznik na wskaznik na mape gry
<i>gridSize</i>	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

<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	wielkosc mapy (NxN)
<i>howManyBoxes</i>	ile na mapie ma sie znajdowac sie pudel
<i>howManyUpgrades</i>	ile na mapie ma sie znajdowac ulepszen

4.13 Dokumentacja pliku mapMemAndGen.h

Funkcje

- void [deallocateMap](#) (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

<i>bitmap</i>	wskaznik na wskaznik na mape gry
<i>gridSize</i>	wielkosc mapy

4.13.1.2 dealocateMap()

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

Dealokuje pamiec mapy o okreslonej wielkosci.

Parametry

<i>bitmap</i>	wskaznik na wskaznik na mape gry
<i>gridSize</i>	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

<i>bitmap</i>	wskaznik na mape gry
<i>gridSize</i>	wielkosc mapy (NxN)
<i>howManyBoxes</i>	ile na mapie ma sie znajdowac sie pudel
<i>howManyUpgrades</i>	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 pudełko na kafelku na którym stoi gracz. Funkcja dostępna tylko w trybie debugu.

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gameTime</i>	czas, w którym gracz ma skonczyc plansze
<i>reset</i>	jesli 1 to nalezy zresetowac wartosci statyczne w funkcji i z niej wyjsc, bez wykonywania reszty jej ciała

4.14.1.2 damagePlayer()

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

Zabija gracza, jeśli ten nie ma aktywnego ulepszenia tarczy. Jeśli ma, to po chwili się je wylacza.

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.14.1.3 decreaseBombFireRange()

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

Obniża zasięg bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.14.1.4 decreaseBombsAvalible()

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

Zmniejsza ilość dostępnych bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.14.1.5 increaseBombFireRange()

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

Zwiększa zasięg bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.14.1.6 increaseBombsAvalible()

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

Zwieksza ilosc dostepnych bomb gracza. Dostepne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.14.1.7 initPlayer()

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

Inicjalizuje gracza, tj. ustawia mu wszystkie dane

Parametry

<i>player</i>	wskaznik na gracza
<i>gameTime</i>	czas, w którym gracz ma skonczyc plansze
<i>gridSize</i>	wielkosc mapy

4.14.1.8 kickSwitch()

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

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

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.14.1.9 moveInADirectionAndKickIfPossibleAndCollectAnUpgrade()

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

Przesuwa gracza, jeśli może. Jeśli na nowym kafelku było ulepszenie, to je zbiera, a jeśli na nowym kafelku była bomba, a gracz ma ulepszenie kopania bomb, to bomba ta zostanie kopnieta

Parametry

<i>dir</i>	kierunek przesunięcia gracza
<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkość mapy

4.14.1.10 openPortal()

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

Otwiera portal, czyli zamienia środkowy kafelek mapy na kafelek portalu

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>gridSize</i>	wielkość mapy

4.14.1.11 placeBomb()

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

Stawia bombę na kafelku na którym stoi gracz

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkość mapy

4.14.1.12 placeBox()

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

Stawia pudełko na kafelku na którym stoi gracz. Funkcja dostępna tylko w trybie debugu.

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkość mapy

4.14.1.13 reInitPlayer()

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

Reinicjalizuje gracza, tj. część danych ustawia mu na nowo

Parametry

<i>player</i>	wskaznik na gracza
<i>gameTime</i>	czas, w którym gracz ma skończyć planszę
<i>gridSize</i>	wielkość mapy

4.14.1.14 resetPlayerStatics()

```
void resetPlayerStatics ( )
```

Wywołuje funkcje zawierające zmienne statyczne tak, by je zresetować

4.14.1.15 shieldSwitch()

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

Włącza lub wyłącza ulepszenie tarczy gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	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

<i>player</i>	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 pudełko na kafelku na którym stoi gracz. Funkcja dostępna tylko w trybie debugu.

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gameTime</i>	czas, w którym gracz ma skończyć planszę
<i>reset</i>	jeśli 1 to należy zresetować wartości statyczne w funkcji i z niej wyjść, bez wykonywania reszty jej ciała

4.15.1.2 `damagePlayer()`

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

Zabija gracza, jeśli ten nie ma aktywnego ulepszenia tarczy. Jeśli ma, to po chwili się je wylacza.

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.15.1.3 decreaseBombFireRange()

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

Obniza zasięg bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.15.1.4 decreaseBombsAvalible()

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

Zmniejsza ilość dostępnych bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.15.1.5 increaseBombFireRange()

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

Zwiększa zasięg bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.15.1.6 increaseBombsAvalible()

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

Zwiększa ilość dostępnych bomb gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	wskaznik na gracza
---------------	--------------------

4.15.1.7 initPlayer()

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

Inicjalizuje gracza, tj. ustawia mu wszystkie dane

Parametry

<i>player</i>	wskaznik na gracza
<i>gameTime</i>	czas, w którym gracz ma skonczyc plansze
<i>gridSize</i>	wielkosc mapy

4.15.1.8 kickSwitch()

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

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

Parametry

<i>player</i>	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

<i>dir</i>	kierunek przesunięcia gracza
<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkość mapy

4.15.1.10 openPortal()

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

Otwiera portal, czyli zamienia środkowy kafelek mapy na kafelek portalu

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>gridSize</i>	wielkość mapy

4.15.1.11 placeBomb()

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

Stawia bombę na kafelku na którym stoi gracz

Parametry

<i>bitmap</i>	wskaznik na mapę gry
<i>player</i>	wskaznik na gracza
<i>gridSize</i>	wielkość mapy

4.15.1.12 placeBox()

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

Stawia pudelko na kafelku na którym stoi gracz. Funkcja dostępna tylko w trybie debugu.

Parametry

<i>bitmap</i>	wskaznik na mape gry
<i>player</i>	wskaznik na gracza
<i>gridSize</i>	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

<i>player</i>	wskaznik na gracza
<i>gameTime</i>	czas, w ktorym gracz ma skonczyc plansze
<i>gridSize</i>	wielkosc mapy

4.15.1.14 resetPlayerStatics()

```
void resetPlayerStatics ( )
```

Wywołuje funkcje zawierające zmienne statyczne tak, by je zresetowac

4.15.1.15 shieldSwitch()

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

Włącza lub wyłącza ulepszenie tarczy gracza. Dostępne tylko w trybie debugowania

Parametry

<i>player</i>	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

<i>player</i>	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 *howManyUpgrades, 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

	<i>argc</i>	Ilość argumentów przekazanych do funkcji main (tj. wraz z wywołaniem programu).
	<i>argv</i>	Dwuwymiarowa tablica przechowująca argumenty przekazane do funkcji main (tj. wraz z wywołaniem programu).
out	<i>howManyEnemies</i>	Ile należy nanieść na plansze przeciwników.
out	<i>howManyBoxes</i>	Ile należy nanieść na plansze pudełek.
out	<i>howManyUpgrades</i>	Ile należy nanieść na plansze ulepszeń.
out	<i>gridSize</i>	Wielkość mapy gry.
out	<i>gameTime</i>	Czas w jakim należy wygrać plansze.
out	<i>debugMode</i>	Czy należy włączyć grę w trybie debugu (dostępne dodatkowe opcje). 1 - tak, 0 - nie.
out	<i>helpSwitch</i>	Czy należy wyświetlić pomoc. 1 - tak, 0 - nie.

Zwraca

Ciąg znaków zawierający wykryte w funkcji błędy w argumentach, pomoc lub ciąg pusty, jeśli żadnych błędów nie było.

4.17 Dokumentacja pliku TestArg.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 *howManyUpgrades, 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 poprawność podanych do funkcji main argumentów i zwraca komunikaty ewentualnych błędów.

Parametry

	<i>argc</i>	Ilosc argumentow przekazanych do funkcji main (tj. wraz z wywołaniem programu).
	<i>argv</i>	Dwuwymiarowa tablica przechowujaca argumenty przekazane do funkcji main (tj. wraz z wywołaniem programu).
out	<i>howManyEnemies</i>	Ile należy naniesc na plansze przeciwnikow.
out	<i>howManyBoxes</i>	Ile należy naniesc na plansze pudelek.
out	<i>howManyUpgrades</i>	Ile należy naniesc na plansze ulepszen.
out	<i>gridSize</i>	Wielkosc mapy gry.
out	<i>gameTime</i>	Czas w jakim należy wygrac plansze.
out	<i>debugMode</i>	Czy należy wlaczyc gre w trybie debugu (dostępne dodatkowe opcje). 1 - tak, 0 - nie.
out	<i>helpSwitch</i>	Czy należy 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

- addEnemy
 - Enemies.c, [27](#)
 - Enemies.h, [31](#)
- allocateMap
 - mapMemAndGen.c, [56](#)
 - mapMemAndGen.h, [57](#)
- Bomb, [5](#)
 - coords, [5](#)
 - explosionTimer, [5](#)
 - fireCross, [6](#)
 - kickDirection, [6](#)
 - kickTimer, [6](#)
 - size, [6](#)
 - statFlag, [6](#)
- bombList
 - Player, [8](#)
- bombState
 - enumsStructsMacros.h, [37](#)
- brokenShieldTimer
 - Player, [8](#)
- check_param
 - TestArg.h, [71](#)
 - TestyArg.c, [72](#)
- checkPlayer
 - player.c, [59](#)
 - player.h, [65](#)
- cleanTile
 - manage bombs.C, [49](#)
 - manage bombs.h, [52](#)
- cleanUp
 - manage bombs.C, [49](#)
 - manage bombs.h, [52](#)
- coords
 - Bomb, [5](#)
 - Enemy, [7](#)
 - Player, [8](#)
- damagePlayer
 - player.c, [60](#)
 - player.h, [65](#)
- deallocateMap
 - mapMemAndGen.c, [57](#)
 - mapMemAndGen.h, [58](#)
- decreaseBombFireRange
 - player.c, [60](#)
 - player.h, [65](#)
- decreaseBombsAvailable
 - player.c, [60](#)
- player.h, [66](#)
- Direction
 - enumsStructsMacros.h, [37](#)
- direction
 - Enemy, [7](#)
- draw.c, [11](#)
 - drawAllEnemies, [11](#)
 - drawBomb, [12](#)
 - drawBombs, [12](#)
 - drawEnemy, [13](#)
 - drawEnemyBlind, [13](#)
 - drawEnemyRandom, [14](#)
 - drawEnemychargerCalm, [13](#)
 - drawEnemychargerUpset, [14](#)
 - drawGrid, [15](#)
 - drawGridBase, [15](#)
 - drawPauseScreen, [16](#)
 - drawPlayer, [16](#)
 - drawPortal, [16](#)
 - drawScoreboard, [16](#)
 - drawSquare, [16](#)
 - drawUpgrade, [17](#)
 - initGrid, [17](#)
 - resetDrawStatics, [18](#)
 - setGILists, [18](#)
 - unSetGIList, [18](#)
- draw.h, [19](#)
 - drawAllEnemies, [19](#)
 - drawBomb, [20](#)
 - drawBombs, [20](#)
 - drawEnemy, [20](#)
 - drawEnemyBlind, [21](#)
 - drawEnemyRandom, [22](#)
 - drawEnemychargerCalm, [21](#)
 - drawEnemychargerUpset, [22](#)
 - drawGrid, [23](#)
 - drawGridBase, [23](#)
 - drawPauseScreen, [23](#)
 - drawPlayer, [23](#)
 - drawPortal, [24](#)
 - drawScoreboard, [24](#)
 - drawSquare, [24](#)
 - drawUpgrade, [25](#)
 - initGrid, [25](#)
 - resetDrawStatics, [26](#)
 - setGILists, [26](#)
 - unSetGIList, [26](#)
- drawAllEnemies
 - draw.c, [11](#)

- draw.h, [19](#)
- drawBomb
 - draw.c, [12](#)
 - draw.h, [20](#)
- drawBombs
 - draw.c, [12](#)
 - draw.h, [20](#)
- drawEnemy
 - draw.c, [13](#)
 - draw.h, [20](#)
- drawEnemyBlind
 - draw.c, [13](#)
 - draw.h, [21](#)
- drawEnemyRandom
 - draw.c, [14](#)
 - draw.h, [22](#)
- drawEnemychargerCalm
 - draw.c, [13](#)
 - draw.h, [21](#)
- drawEnemychargerUpset
 - draw.c, [14](#)
 - draw.h, [22](#)
- drawGrid
 - draw.c, [15](#)
 - draw.h, [23](#)
- drawGridBase
 - draw.c, [15](#)
 - draw.h, [23](#)
- drawPauseScreen
 - draw.c, [16](#)
 - draw.h, [23](#)
- drawPlayer
 - draw.c, [16](#)
 - draw.h, [23](#)
- drawPortal
 - draw.c, [16](#)
 - draw.h, [24](#)
- drawScoreboard
 - draw.c, [16](#)
 - draw.h, [24](#)
- drawSquare
 - draw.c, [16](#)
 - draw.h, [24](#)
- drawUpgrade
 - draw.c, [17](#)
 - draw.h, [25](#)
- dropEnemies
 - Enemies.c, [27](#)
 - Enemies.h, [32](#)
- Enemies.c, [27](#)
 - addEnemy, [27](#)
 - dropEnemies, [27](#)
 - initEnemies, [28](#)
 - manageEnemiesAndScore, [28](#)
 - moveBlind, [29](#)
 - moveCharger, [29](#)
 - moveEnemy, [30](#)
 - moveRandom, [30](#)
 - pointTowardsPlayerIfPossible, [30](#)
 - resetEnemyStatics, [31](#)
- Enemies.h, [31](#)
 - addEnemy, [31](#)
 - dropEnemies, [32](#)
 - initEnemies, [32](#)
 - manageEnemiesAndScore, [32](#)
 - moveBlind, [33](#)
 - moveCharger, [33](#)
 - moveEnemy, [34](#)
 - moveRandom, [34](#)
 - pointTowardsPlayerIfPossible, [35](#)
 - resetEnemyStatics, [35](#)
- Enemy, [6](#)
 - coords, [7](#)
 - direction, [7](#)
 - rottingTime, [7](#)
 - status, [7](#)
 - type, [7](#)
- EnemyType
 - enumsStructsMacros.h, [37](#)
- enumsStructsMacros.h, [35](#)
 - bombState, [37](#)
 - Direction, [37](#)
 - EnemyType, [37](#)
 - PauseScreenType, [38](#)
 - Status, [38](#)
 - tileType, [39](#)
 - upgrades, [39](#)
- error_callback
 - main.c, [45](#)
- explodeBombAddPoints
 - manage bombs.C, [49](#)
 - manage bombs.h, [54](#)
- explosionTimer
 - Bomb, [5](#)
- fireCross
 - Bomb, [6](#)
- fireLayers
 - tile, [10](#)
- fonts.h, [41](#)
 - sign, [41](#)
 - translateWord, [41](#)
 - writeDynamicData, [42](#)
 - writePauseMessage, [42](#)
 - writePauseMessageScore, [42](#)
 - writeStaticData, [43](#)
- generateMap
 - mapMemAndGen.c, [57](#)
 - mapMemAndGen.h, [58](#)
- getBombID
 - manage bombs.C, [50](#)
 - manage bombs.h, [54](#)
- getHighscore
 - highscoreFile.c, [43](#)
 - highscoreFile.h, [44](#)

- highscoreFile.c, [43](#)
 - getHighscore, [43](#)
 - setHighscore, [43](#)
- highscoreFile.h, [44](#)
 - getHighscore, [44](#)
 - setHighscore, [44](#)
- increaseBombFireRange
 - player.c, [60](#)
 - player.h, [66](#)
- increaseBombsAvailible
 - player.c, [61](#)
 - player.h, [66](#)
- init
 - main.c, [45](#)
- initEnemies
 - Enemies.c, [28](#)
 - Enemies.h, [32](#)
- initGrid
 - draw.c, [17](#)
 - draw.h, [25](#)
- initPlayer
 - player.c, [61](#)
 - player.h, [67](#)
- key_callback
 - main.c, [46](#)
- kickDirection
 - Bomb, [6](#)
- kickSwitch
 - player.c, [61](#)
 - player.h, [67](#)
- kickTimer
 - Bomb, [6](#)
- main.c, [45](#)
 - error_callback, [45](#)
 - init, [45](#)
 - key_callback, [46](#)
 - pauseGame, [47](#)
 - resetTimes, [47](#)
 - setDefaultslfNecessary, [47](#)
 - unPauseGame, [48](#)
- manage bombs.C, [48](#)
 - cleanTile, [49](#)
 - cleanUp, [49](#)
 - explodeBombAddPoints, [49](#)
 - getBombID, [50](#)
 - moveBombData, [50](#)
 - moveCheckExplodeAndCleanUpBombs, [51](#)
 - moveTile, [51](#)
 - spreadFire, [51](#)
- manage bombs.h, [52](#)
 - cleanTile, [52](#)
 - cleanUp, [52](#)
 - explodeBombAddPoints, [54](#)
 - getBombID, [54](#)
 - moveBombData, [54](#)
 - moveCheckExplodeAndCleanUpBombs, [55](#)
 - moveTile, [55](#)
 - spreadFire, [56](#)
- manageEnemiesAndScore
 - Enemies.c, [28](#)
 - Enemies.h, [32](#)
- mapMemAndGen.c, [56](#)
 - allocateMap, [56](#)
 - deallocateMap, [57](#)
 - generateMap, [57](#)
- mapMemAndGen.h, [57](#)
 - allocateMap, [57](#)
 - deallocateMap, [58](#)
 - generateMap, [58](#)
- moveBlind
 - Enemies.c, [29](#)
 - Enemies.h, [33](#)
- moveBombData
 - manage bombs.C, [50](#)
 - manage bombs.h, [54](#)
- moveCharger
 - Enemies.c, [29](#)
 - Enemies.h, [33](#)
- moveCheckExplodeAndCleanUpBombs
 - manage bombs.C, [51](#)
 - manage bombs.h, [55](#)
- moveEnemy
 - Enemies.c, [30](#)
 - Enemies.h, [34](#)
- moveInADirectionAndKickIfPossibleAndCollectAn→
Upgrade
 - player.c, [62](#)
 - player.h, [67](#)
- moveRandom
 - Enemies.c, [30](#)
 - Enemies.h, [34](#)
- moveTile
 - manage bombs.C, [51](#)
 - manage bombs.h, [55](#)
- newestBomb
 - Player, [8](#)
- oldestBomb
 - Player, [9](#)
- openPortal
 - player.c, [62](#)
 - player.h, [68](#)
- pauseGame
 - main.c, [47](#)
- PauseScreenType
 - enumsStructsMacros.h, [38](#)
- placeBomb
 - player.c, [62](#)
 - player.h, [68](#)
- placeBox
 - player.c, [63](#)
 - player.h, [68](#)
- Player, [8](#)

- bombList, 8
- brokenShieldTimer, 8
- coords, 8
- newestBomb, 8
- oldestBomb, 9
- score, 9
- status, 9
- timeLeft, 9
- upgrades, 9
- player.c, 59
 - checkPlayer, 59
 - damagePlayer, 60
 - decreaseBombFireRange, 60
 - decreaseBombsAvailable, 60
 - increaseBombFireRange, 60
 - increaseBombsAvailable, 61
 - initPlayer, 61
 - kickSwitch, 61
 - moveInADirectionAndKickIfPossibleAndCollect↔
 - AnUpgrade, 62
 - openPortal, 62
 - placeBomb, 62
 - placeBox, 63
 - reInitPlayer, 63
 - resetPlayerStatics, 63
 - shieldSwitch, 63
 - waitForGame_over, 64
 - waitForNextStageAndSumScore, 64
- player.h, 64
 - checkPlayer, 65
 - damagePlayer, 65
 - decreaseBombFireRange, 65
 - decreaseBombsAvailable, 66
 - increaseBombFireRange, 66
 - increaseBombsAvailable, 66
 - initPlayer, 67
 - kickSwitch, 67
 - moveInADirectionAndKickIfPossibleAndCollect↔
 - AnUpgrade, 67
 - openPortal, 68
 - placeBomb, 68
 - placeBox, 68
 - reInitPlayer, 70
 - resetPlayerStatics, 70
 - shieldSwitch, 70
 - waitForGame_over, 70
 - waitForNextStageAndSumScore, 71
- pointTowardsPlayerIfPossible
 - Enemies.c, 30
 - Enemies.h, 35
- reInitPlayer
 - player.c, 63
 - player.h, 70
- resetDrawStatics
 - draw.c, 18
 - draw.h, 26
- resetEnemyStatics
 - Enemies.c, 31
 - Enemies.h, 35
- resetPlayerStatics
 - player.c, 63
 - player.h, 70
- resetTimes
 - main.c, 47
- rottingTime
 - Enemy, 7
- score
 - Player, 9
- setDefaultIfNecessary
 - main.c, 47
- setGILists
 - draw.c, 18
 - draw.h, 26
- setHighscore
 - highscoreFile.c, 43
 - highscoreFile.h, 44
- shieldSwitch
 - player.c, 63
 - player.h, 70
- sign
 - fonts.h, 41
- size
 - Bomb, 6
- spreadFire
 - manage bombs.C, 51
 - manage bombs.h, 56
- statFlag
 - Bomb, 6
- Status
 - enumsStructsMacros.h, 38
- status
 - Enemy, 7
 - Player, 9
- TestArg.h, 71
 - check_param, 71
- TestyArg.c, 72
 - check_param, 72
- tile, 9
 - fireLayers, 10
 - type, 10
 - upgrade, 10
- tileType
 - enumsStructsMacros.h, 39
- timeLeft
 - Player, 9
- translateWord
 - fonts.h, 41
- type
 - Enemy, 7
 - tile, 10
- unPauseGame
 - main.c, 48
- unSetGIList
 - draw.c, 18

- draw.h, [26](#)
- upgrade
 - tile, [10](#)
- upgrades
 - enumsStructsMacros.h, [39](#)
 - Player, [9](#)
- waitForGame_over
 - player.c, [64](#)
 - player.h, [70](#)
- waitForNextStageAndSumScore
 - player.c, [64](#)
 - player.h, [71](#)
- writeDynamicData
 - fonts.h, [42](#)
- writePauseMessage
 - fonts.h, [42](#)
- writePauseMessageScore
 - fonts.h, [42](#)
- writeStaticData
 - fonts.h, [43](#)
- x
 - XY, [10](#)
- XY, [10](#)
 - x, [10](#)
 - y, [10](#)
- y
 - XY, [10](#)