Space Invaders

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Game Namespace Reference

Namespace with logistic of the game.

Functions

- void **handle_events** (sf::Window &window, sf::RenderWindow &windows, **Textures** &textures, **Clock**↩ **Display** &clock_disp, **Info** &info, **Save** &save, **MenuIG** &menuig, **Load** &load, **Menu** &menu, **Score**↩ **Display** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **PlayerLaser** &player_↩ laser, **Spaceship** &ship, **Explosions** &explosions, unsigned &wave_on)

  Events(mainly window close)
- void **real_time_key** (**PlayerLaser** &player_laser, **Spaceship** &ship)

  Real-time keyboard input.
- void **update_objects** (sf::RenderWindow &window, **Stars** &stars, **MenuIG** &menuig, **Load** &load, **Save** &save, **Menu** &menu, **Textures** &textures, **Spaceship** &ship, **PlayerLaser** &player_laser, **Invader**↩ **Formation** &invaders, **LivesDisplay** &lives_disp, **Explosions** &explosions, unsigned &game_score, unsigned &wave_on, unsigned &minutes, unsigned &seconds, unsigned &miliseconds)

  Update all game objects.
- void **draw_player_laser** (sf::RenderWindow &window, **PlayerLaser** &laser)

  Draw player lasers.
- void **draw_text** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, sf::Color color=sf::Color::White, unsigned size=24)

  Draw text on screen.
- void **draw_objects** (sf::RenderWindow &window, **Stars** &stars, **ClockDisplay** &clock_disp, **MenuIG** &menuig, **Load** &load, **Save** &save, **Info** &info, **Menu** &menu, **ScoreDisplay** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  Draw objects on screen.
- void **handle_player_kill** (**InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Explosions** &explosions)

  An event after player was killed (mainly pause)
- void **setup_wave** (sf::RenderWindow &window, **Textures** &textures, **InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Spaceship** &ship, **Explosions** &explosions, unsigned &wave_on, bool start↩ _game=false)

  Setup new wave.

- void **draw_wave** (**InvaderFormation** &invaders)

  Draw wave on screen.
- void **handle_game_over** (sf::RenderWindow &window, **ClockDisplay** &clock_disp, **Menu** &menu, **Score↩ Display** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  *An event after **Game** (p. 7) Over.*
- void **reset_game** (**Menu** &menu, **ClockDisplay** &clock_disp, **ScoreDisplay** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  Reset all objects.
- void **goto_menu** (**Menu** &menu, **ClockDisplay** &clock_disp, **ScoreDisplay** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  Go to main menu.
- void **goto_menuig** (**MenuIG** &menuig)

  Go to menu in game.
- void **goto_info** (**Info** &info)

  See info about Invaders.
- void **goto_loading** (**Load** &load)

  Go to State for Loading game.
- void **goto_saving** (**Save** &save)

  Go to State for Saving game.
- void **play_game** (sf::RenderWindow &window, **Textures** &textures, **InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Spaceship** &ship, **Explosions** &explosions, unsigned &wave_on)

  *Play New **Game** (p. 7).*

Variables

- bool **life_awarded** = false

  Variable to see if life was awarded.

4.1.1 Detailed Description

Namespace with logistic of the game.

Name: **game.h** (p. 94) Purpose: Declaration of namespace **Game** (p. 7)

Author

Fenris

Version

1.03 14/05/2017

4.1.2 Function Documentation

4.1.2.1 draw_objects()

```
void Game::draw_objects (
             sf::RenderWindow & window,
              Stars & stars,
              ClockDisplay & clock_disp,
              MenuIG & menuig,
              Load & load,
              Save & save,
              Info & info,
              Menu & menu,
              ScoreDisplay & score_disp,
              LivesDisplay & lives_disp,
              InvaderFormation & invaders,
              Spaceship & ship,
              PlayerLaser & playerlaser,
              Explosions & explosions,
             unsigned & wave_on )
```

Draw objects on screen.

Draw all the objects we want.

4.1.2.2 draw_player_laser()

```
void Game::draw_player_laser (
             sf::RenderWindow & window,
              PlayerLaser & laser )
```

Draw player lasers.

Draw the player's laser beam.

4.1.2.3 draw_text()

```
void Game::draw_text (
             sf::RenderWindow & window,
             const std::string msg,
             const unsigned x,
             const unsigned y,
             sf::Color color = sf::Color::White,
             unsigned size = 24 )
```

Draw text on screen.

It's a simple method to get text on the screen.

4.1.2.4 draw_wave()

```
void Game::draw_wave (
              InvaderFormation & invaders )
```

Draw wave on screen.

Draw Invaders one by one.

4.1.2.5 goto_info()

```
void Game::goto_info (
              Info & info )
```

See info about Invaders.

Go to info.

4.1.2.6 goto_loading()

```
void Game::goto_loading (
              Load & load )
```

Go to State for Loading game.

Go to loading.

4.1.2.7 goto_menu()

```
void Game::goto_menu (
              Menu & menu,
              ClockDisplay & clock_disp,
              ScoreDisplay & score_disp,
              LivesDisplay & lives_disp,
              InvaderFormation & invaders,
              Spaceship & ship,
              PlayerLaser & playerlaser,
              Explosions & explosions,
             unsigned & wave_on )
```

Go to main menu.

Go to menu and reset all your progress.

4.1.2.8 goto_menuig()

```
void Game::goto_menuig (
              MenuIG & menuig )
```

Go to menu in game.

4.1.2.9 goto_saving()

```
void Game::goto_saving (
              Save & save )
```

Go to State for Saving game.

Go to saving.

4.1.2.10 handle_events()

```
void Game::handle_events (
             sf::Window & window,
             sf::RenderWindow & windows,
              Textures & textures,
              ClockDisplay & clock_disp,
              Info & info,
              Save & save,
              MenuIG & menuig,
              Load & load,
              Menu & menu,
              ScoreDisplay & score_disp,
              LivesDisplay & lives_disp,
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              Spaceship & ship,
              Explosions & explosions,
             unsigned & wave_on )
```

Events(mainly window close)

It's just controls for getting to menu, or to close the game. Which key?

If you are in state MENU

If you are in state LOAD

If you are in state SAVE

If you are in state MENUIG

Which button?

4.1.2.11 handle_game_over()

```
void Game::handle_game_over (
             sf::RenderWindow & window,
              ClockDisplay & clock_disp,
              Menu & menu,
              ScoreDisplay & score_disp,
              LivesDisplay & lives_disp,
              InvaderFormation & invaders,
              Spaceship & ship,
              PlayerLaser & playerlaser,
              Explosions & explosions,
             unsigned & wave_on )
```

An event after **Game** (p. 7) Over.

How to handle GAME OVER? Let's show you how I spell LOSER, letter by letter

After this see for 5 seconds what have you done, then I will show you menu

4.1.2.12 handle_player_kill()

```
void Game::handle_player_kill (
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              Explosions & explosions )
```

An event after player was killed (mainly pause)

Stop everything because you were killed.

4.1.2.13 play_game()

```
void Game::play_game (
             sf::RenderWindow & window,
              Textures & textures,
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              Spaceship & ship,
              Explosions & explosions,
             unsigned & wave_on )
```

Play New **Game** (p. 7).

State: Play game.

4.1.2.14 real_time_key()

```
void Game::real_time_key (
              PlayerLaser & player_laser,
              Spaceship & ship )
```

Real-time keyboard input.

Controls to move your ship, play game or fire lasers. Move your ship out of my property!

I'M A' FIRIN' MAH LAZER!!

4.1.2.15 reset_game()

```
void Game::reset_game (
              Menu & menu,
              ClockDisplay & clock_disp,
              ScoreDisplay & score_disp,
              LivesDisplay & lives_disp,
              InvaderFormation & invaders,
              Spaceship & ship,
              PlayerLaser & playerlaser,
              Explosions & explosions,
             unsigned & wave_on )
```

Reset all objects.

Reset all you have already done.

4.1.2.16 setup_wave()

```
void Game::setup_wave (
             sf::RenderWindow & window,
              Textures & textures,
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              Spaceship & ship,
              Explosions & explosions,
             unsigned & wave_on,
             bool start_game = false )
```

Setup new wave.

Set the new wave.

4.1.2.17 update_objects()

```
void Game::update_objects (
             sf::RenderWindow & window,
              Stars & stars,
              MenuIG & menuig,
              Load & load,
              Save & save,
              Menu & menu,
              Textures & textures,
              Spaceship & ship,
              PlayerLaser & player_laser,
              InvaderFormation & invaders,
              LivesDisplay & lives_disp,
              Explosions & explosions,
             unsigned & game_score,
             unsigned & wave_on,
             unsigned & minutes,
             unsigned & seconds,
             unsigned & miliseconds )
```

Update all game objects.

Up to date with all the objects on the screen.

4.1.3 Variable Documentation

4.1.3.1 life_awarded

```
bool Game::life_awarded = false
```

Variable to see if life was awarded.

Name: **game.cpp** (p. 94) Purpose: Namespace **Game** (p. 7)

Author

Fenris

Version

1.03a 14/05/2017

4.2 Globals Namespace Reference

Enumerations

- enum **States** {
  MENU, **MENUIG**, **PLAY**, **WAVE_SETUP**,
  PLAYER_KILLED, **SAVE**, **LOAD**, **INFO**,
  GAME_OVER }
  States the game can be in.

Variables

- const std::string **SCREEN_TITLE** = "Space Invaders"
  Title of Window.
- constexpr unsigned **SCREEN_WIDTH** = 1366
  Width of Window.
- constexpr unsigned **SCREEN_HEIGHT** = 768
  Height of Window.
- constexpr unsigned **FRAME_RATE** = 60
  Frame rate.
- const sf::Color **BG_COLOR** = sf::Color(8,8,16)
  Color of the background.
- const std::string **SPRITES_PATH** = "sprites/"
  *Path to get sprites for **Textures** (p. 78).*
- const std::string **FONTS_PATH** = "fonts/"
  Path to get font.
- const std::string **SAVES_PATH** = "saves/"
  Path to get files for load and where will be saved new files.
- const std::string **LEVELS_PATH** = "levels/"
  Path to get files for loading levels.
- **States** **GAME_STATE** = Globals::States::MENU
  Variable to know in which state we are now.
- **States** **PREVIOUS_STATE** = Globals::States::MENU
  Variable to know in which state we were before.

4.2.1 Detailed Description

Name: **globals.h** (p. 96) Purpose: Declaration of namespace **Globals** (p. 14)

Author

Fenris

Version

1.03a 14/05/2017

4.2.2 Enumeration Type Documentation

4.2.2.1 States

`enum` **`Globals::States`**

States the game can be in.

Enumerator

MENU	
MENUIG	
PLAY	
WAVE_SETUP	
PLAYER_KILLED	
SAVE	
LOAD	
INFO	
GAME_OVER	

4.2.3 Variable Documentation

4.2.3.1 BG_COLOR

```
const sf::Color Globals::BG_COLOR = sf::Color(8,8,16)
```

Color of the background.

4.2.3.2 FONTS_PATH

```
const std::string Globals::FONTS_PATH = "fonts/"
```

Path to get font.

4.2.3.3 FRAME_RATE

```
constexpr unsigned Globals::FRAME_RATE = 60
```

Frame rate.

4.2.3.4 GAME_STATE

```
Globals::States Globals::GAME_STATE = Globals::States::MENU
```

Variable to know in which state we are now.

Name: **globals.cpp** (p. 96) Purpose: Namespace **Globals** (p. 14)

Author

Fenris

Version

0.16 01/05/2017

4.2.3.5 LEVELS_PATH

```
const std::string Globals::LEVELS_PATH = "levels/"
```

Path to get files for loading levels.

4.2.3.6 PREVIOUS_STATE

```
Globals::States Globals::PREVIOUS_STATE = Globals::States::MENU
```

Variable to know in which state we were before.

4.2.3.7 SAVES_PATH

```
const std::string Globals::SAVES_PATH = "saves/"
```

Path to get files for load and where will be saved new files.

4.2.3.8 SCREEN_HEIGHT

```
constexpr unsigned Globals::SCREEN_HEIGHT = 768
```

Height of Window.

4.2.3.9 SCREEN_TITLE

```
const std::string Globals::SCREEN_TITLE = "Space Invaders"
```

Title of Window.

4.2.3.10 SCREEN_WIDTH

```
constexpr unsigned Globals::SCREEN_WIDTH = 1366
```

Width of Window.

4.2.3.11 SPRITES_PATH

```
const std::string Globals::SPRITES_PATH = "sprites/"
```

Path to get sprites for **Textures** (p. 78).

4.3 Rand Namespace Reference

Namespace of **Rand** (p. 17).

Functions

- unsigned **random** (const unsigned low, const unsigned high)
  Randomizing.

4.3.1 Detailed Description

Namespace of **Rand** (p. 17).

Name: **rand.h** (p. 103) Purpose: Declaration of namespace **Rand** (p. 17)

Author

Fenris

Version

0.18 01/05/2017

4.3.2 Function Documentation

4.3.2.1 random()

```
unsigned Rand::random (
            const unsigned low,
            const unsigned high )
```

Randomizing.

Parameters

low	The bottom line for randomizing
high	The upper line for randomizing

Name: **rand.cpp** (p. 103) Purpose: Namespace **Rand** (p. 17)

Author

Fenris

Version

0.18a 01/05/2017

Chapter 5

Class Documentation

5.1 Bonus Class Reference

Class which consists of functions for displaying bonuses.

`#include <Bonus.h>`

Public Types

- enum **BonusType** {
  BONUS_1, **BONUS_2**, **BONUS_3**, **BONUS_4**,
  BONUS_5, **BONUS_6**, **BONUS_7**, **BONUS_8** }

  Container for types of bonuses.

Public Member Functions

- **Bonus** (const unsigned x, const unsigned y, **Invader** &owner, **Textures** &textures, const **BonusType** type)

  *Default constructor for **Bonus** (p. 19).*
- ~**Bonus** ()

  *Default destructor for **Bonus** (p. 19).*
- bool **isHit** () const

  *Check if **Bonus** (p. 19) get status: hit.*
- void **setHit** ()

  *Set that **Bonus** (p. 19) got status: hit.*
- void **getBonus** (const **BonusType** type, **LivesDisplay** &lives, unsigned &game_score, **PlayerLaser** &laser)

  *Get bonus from **Bonus** (p. 19) you caught.*
- **BonusType** **getType** ()

  *Check what type of **Bonus** (p. 19) is this.*
- unsigned **getX** () const

  *Get X Position of **Bonus** (p. 19) on scene.*
- unsigned **getY** () const

  *Get Y Position of **Bonus** (p. 19) on scene.*
- void **move** ()

  *Move **Bonus** (p. 19).*
- void **draw** (sf::RenderWindow &window)

  *Draw sprite of **Bonus** (p. 19).*
- bool **checkCollide** (const unsigned x, const unsigned y) const

  Check collisions with players ship.
- bool **checkCollide** (const sf::FloatRect rect) const

  Check collisions with players ship.

5.1.1 Detailed Description

Class which consists of functions for displaying bonuses.

Name: **Bonus.h** (p. 93) Purpose: Declaration of class **Bonus** (p. 19)

Author

Fenris

Version

0.81b 03/05/2017

5.1.2 Member Enumeration Documentation

5.1.2.1 BonusType

`enum` **Bonus::BonusType**

Container for types of bonuses.

Enumerator

BONUS_1	
BONUS_2	
BONUS_3	
BONUS_4	
BONUS_5	
BONUS_6	
BONUS_7	
BONUS_8	

5.1.3 Constructor & Destructor Documentation

5.1.3.1 Bonus()

```
Bonus::Bonus (
            const unsigned x,
            const unsigned y,
             Invader & owner,
             Textures & textures,
            const  BonusType type )
```

Default constructor for **Bonus** (p. 19).

Parameters

x	X Position of **Bonus** (p. 19) on scene
y	Y Position of **Bonus** (p. 19) on scene
owner	**Invader** (p. 33) from which **Bonus** (p. 19) was dropped out
textures	**Textures** (p. 78) of Bonuses
type	Type of **Bonus** (p. 19)

Name: **Bonus.cpp** (p. 93) Purpose: Class **Bonus** (p. 19)

Author

Fenris

Version

0.88a 03/05/2017

5.1.3.2 ∼Bonus()

```
Bonus::∼Bonus ( )  [inline]
```

Default destructor for **Bonus** (p. 19).

5.1.4 Member Function Documentation

5.1.4.1 checkCollide() [1/2]

```
bool Bonus::checkCollide (
            const unsigned x,
            const unsigned y ) const
```

Check collisions with players ship.

Parameters

x	X Position of players ship
y	Y Position of players ship

5.1.4.2 checkCollide() [2/2]

```
bool Bonus::checkCollide (
            const sf::FloatRect rect ) const
```

Check collisions with players ship.

Parameters

rect	Rectangle of players ship

5.1.4.3 draw()

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

Draw sprite of **Bonus** (p. 19).

Parameters

window	Where it should be drawn

5.1.4.4 getBonus()

```
void Bonus::getBonus (
            const  BonusType type,
             LivesDisplay & lives,
            unsigned & game_score,
             PlayerLaser & laser )
```

Get bonus from **Bonus** (p. 19) you caught.

Parameters

type	Type of **Bonus** (p. 19)
lives	For operations on lives
game_score	For operations on game score
laser	For operations on players laser

5.1.4.5 getType()

```
BonusType Bonus::getType ( )  [inline]
```

Check what type of **Bonus** (p. 19) is this.

Returns

Type of bonus

5.1.4.6 getX()

```
unsigned Bonus::getX ( ) const  [inline]
```

Get X Position of **Bonus** (p. 19) on scene.

5.1.4.7 getY()

```
unsigned Bonus::getY ( ) const  [inline]
```

Get Y Position of **Bonus** (p. 19) on scene.

5.1.4.8 isHit()

```
bool Bonus::isHit ( ) const  [inline]
```

Check if **Bonus** (p. 19) get status: hit.

Returns

Bool for status (is hit?)

5.1.4.9 move()

```
void Bonus::move ( )
```

Move **Bonus** (p. 19).

5.1.4.10 setHit()

```
void Bonus::setHit ( )  [inline]
```

Set that **Bonus** (p. 19) got status: hit.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Bonus.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Bonus.cpp**

5.2 ClockDisplay Class Reference

Class for displaying clock.

`#include <ClockDisplay.h>`

Public Member Functions

- **ClockDisplay** (unsigned &minutes, unsigned &seconds, unsigned &miliseconds)

  *Default constructor for **ClockDisplay** (p. 24).*
- void **drawClock** (sf::RenderWindow &window, const unsigned minutes, const unsigned seconds, const unsigned miliseconds, const unsigned x, const unsigned y)

  Get text to display on scene.
- void **draw** (sf::RenderWindow &window)

  Draw Clock(text) on scene.
- void **reset** ()

  Reset minutes, seconds and miliseconds to drop them down to 0.

5.2.1 Detailed Description

Class for displaying clock.

Name: **ClockDisplay.h** (p. 93) Purpose: Declaration of class **ClockDisplay** (p. 24)

Author

Fenris

Version

1.05 21/05/2017

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ClockDisplay()

```
ClockDisplay::ClockDisplay (
            unsigned & minutes,
            unsigned & seconds,
            unsigned & miliseconds )
```

Default constructor for **ClockDisplay** (p. 24).

Parameters

minutes	Variable for minutes to display on Clock
seconds	Variable for seconds to display on Clock
miliseconds	Variable for miliseconds to display on Clock

Name: **ClockDisplay.cpp** (p. 93) Purpose: Class **ClockDisplay** (p. 24)

Author

Fenris

Version

1.05a 03/05/2017

5.2.3 Member Function Documentation

5.2.3.1 draw()

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

Draw Clock(text) on scene.

5.2.3.2 drawClock()

```
void ClockDisplay::drawClock (
            sf::RenderWindow & window,
            const unsigned minutes,
            const unsigned seconds,
            const unsigned miliseconds,
            const unsigned x,
            const unsigned y )
```

Get text to display on scene.

Parameters

window	Where it should be drawn
minutes	Variable for minutes to display on Clock
seconds	Variable for seconds to display on Clock
miliseconds	Variable for miliseconds to display on Clock
x	X Position of **ClockDisplay** (p. 24) on scene
y	Y Position of **ClockDisplay** (p. 24) on scene

5.2.3.3 reset()

```
void ClockDisplay::reset ( )
```

Reset minutes, seconds and miliseconds to drop them down to 0.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **ClockDisplay.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **ClockDisplay.cpp**

5.3 Explosion Class Reference

Class for one explosion.

```
#include <Explosions.h>
```

Public Member Functions

- **Explosion** (**Textures** &textures, sf::Color color, const unsigned x, const unsigned y)

  Is explosion showing?
- bool **isShowing** () const

  *Get bool of showing (is **Explosion** (p. 26) showing?)*
- void **update** ()

  Update explosion till it's showing.
- void **draw** (sf::RenderWindow &window)

  *Draw **Explosion** (p. 26) on scene.*

5.3.1 Detailed Description

Class for one explosion.

Name: **Explosions.h** (p. 94) Purpose: Declaration of classes **Explosion** (p. 26) and **Explosions** (p. 28)

Author

Fenris

Version

0.98 03/05/2017

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Explosion()

```
Explosion::Explosion (
              Textures & textures,
             sf::Color color,
             const unsigned x,
             const unsigned y )
```

Is explosion showing?

Explosion (p. 26) class.

Default constructor for **Explosion** (p. 26)

Parameters

textures	Texture of **Explosion** (p. 26)
color	Color of **Explosion** (p. 26)
x	X Position of **Explosion** (p. 26) on scene
y	Y Position of **Explosion** (p. 26) on scene

Name: **Explosions.cpp** (p. 94) Purpose: Classes **Explosion** (p. 26) and **Explosions** (p. 28)

Author

Fenris

Version

0.65a 03/05/2017

5.3.3 Member Function Documentation

5.3.3.1 draw()

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

Draw **Explosion** (p. 26) on scene.

Parameters

window	Where it should be drawn

5.3.3.2 isShowing()

```
bool Explosion::isShowing ( ) const  [inline]
```

Get bool of showing (is **Explosion** (p. 26) showing?)

Returns

Bool if **Explosion** (p. 26) is showing or no?

5.3.3.3 update()

```
void Explosion::update ( )
```

Update explosion till it's showing.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Explosions.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Explosions.cpp**

5.4 Explosions Class Reference

Class for all explosions.

```
#include <Explosions.h>
```

Public Member Functions

- **Explosions** (**Textures** &textures)
  *Default constructor for **Explosions** (p. 28).*
- ~**Explosions** ()
  *Default destructor for **Explosions** (p. 28).*
- void **reset** ()
  Clear vector to get it clean.
- void **update** ()
  *If **Explosion** (p. 26) is not showing -> Delete it.*
- void **draw** (sf::RenderWindow &window)
  Draw explosions on scene.
- void **newExplosion** (sf::Color color, const unsigned x, const unsigned y)
  Push new explosion to vector.

5.4.1 Detailed Description

Class for all explosions.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Explosions()

```
Explosions::Explosions (
              Textures & textures )
```

Default constructor for **Explosions** (p. 28).

Explosions (p. 28) class.

Parameters

textures	Texture of **Explosions** (p. 28)

5.4.2.2 ~Explosions()

```
Explosions::~Explosions ( )
```

Default destructor for **Explosions** (p. 28).

5.4.3 Member Function Documentation

5.4.3.1 draw()

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

Draw explosions on scene.

Parameters

window	Where they should be drawn

5.4.3.2 newExplosion()

```
void Explosions::newExplosion (
             sf::Color color,
```

```
        const unsigned x,
        const unsigned y )
```

Push new explosion to vector.

Parameters

color	Color of new **Explosion** (p. 26)
x	X Position for new **Explosion** (p. 26)
y	Y Position for new **Explosion** (p. 26)

5.4.3.3 reset()

```
void Explosions::reset ( )
```

Clear vector to get it clean.

5.4.3.4 update()

```
void Explosions::update ( )
```

If **Explosion** (p. 26) is not showing -> Delete it.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Explosions.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Explosions.cpp**

5.5 Info Class Reference

Class of State INFO.

```
#include <Info.h>
```

Public Member Functions

- **Info** (**Textures** &textures, **ScoreDisplay** &score_disp, **ClockDisplay** &clock_disp)

  *Default constructor for **Info** (p. 30).*
- void **reset** ()

  *Reset **Info** (p. 30) to get specific type of drawing.*
- void **drawLine** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, const unsigned wait, const unsigned line_num, sf::Sprite *sprite=nullptr, sf::Color color=sf::Color::White, unsigned size=24)

  Draws a string with pauses after each character (wait)
- void **draw** (sf::RenderWindow &window)

  *Draw **Info** (p. 30).*

5.5.1 Detailed Description

Class of State INFO.

Name: **Info.h** (p. 97) Purpose: Declaration of class **Info** (p. 30)

Author

Fenris

Version

1.01 14/05/2017

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Info()

```
Info::Info (
              Textures & textures,
              ScoreDisplay & score_disp,
              ClockDisplay & clock_disp )
```

Default constructor for **Info** (p. 30).

Parameters

textures	**Textures** (p. 78) which will be displayed next to score
score_disp	Score displayed on scene
clock_disp	Clock displayed on scene

Name: **Info.cpp** (p. 97) Purpose: Class **Info** (p. 30)

Author

Fenris

Version

1.02a 14/05/2017

5.5.3 Member Function Documentation

5.5.3.1 draw()

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

Draw **Info** (p. 30).

Parameters

window	Where should it be displayed

5.5.3.2 drawLine()

```
void Info::drawLine (
            sf::RenderWindow & window,
            const std::string msg,
            const unsigned x,
            const unsigned y,
            const unsigned wait,
            const unsigned line_num,
            sf::Sprite * sprite = nullptr,
            sf::Color color = sf::Color::White,
            unsigned size = 24 )
```

Draws a string with pauses after each character (wait)

Parameters

window	Where should it be displayed
msg	What message should be displayed
x	X Position where it should be displayed on scene
y	Y Position where it should be displayed on scene
wait	How much time pause should take
line_num	What line is it on?
sprite	What sprite should be displayed next to text
color	In which color text should be displayed?
size	What size of font should text has?

Don't do anything until info isn't done with previous line

If it's okey it will drew next line

Draw the line letter by letter

5.5.3.3 reset()

```
void Info::reset ( )
```

Reset **Info** (p. 30) to get specific type of drawing.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Info.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Info.cpp**

5.6 Invader Class Reference

Class for **Invader** (p. 33).

```
#include <Invader.h>
```

Public Types

- enum **InvaderType** {
  INVADER_1, **INVADER_2**, **INVADER_3**, **INVADER_4**,
  UFO, **CREEPER**, **TOUGH**, **BOSS** }

  Container for types of Invaders.

Public Member Functions

- **Invader** (**Textures** &textures, const **InvaderType** type, int **isVisible**)

  *Default constructor for **Invader** (p. 33).*
- sf::Sprite & **getSprite** ()

  *Get **Invader** (p. 33)'s sprite.*
- unsigned **getScoreValue** () const

  Get how mush score user will get.
- bool **isDead** () const

  *Check if **Invader** (p. 33) is dead.*
- bool **isExploding** () const

  *Check if **Invader** (p. 33) is exploding.*
- bool **isVisible** () const

  *Check if **Invader** (p. 33) is visible.*
- int **getMoveDir** () const

  Check movement direction.
- int **getLives** () const

  *Check how many lives **Invader** (p. 33) has.*
- void **setLives** (int life)

  *Set how many lives should **Invader** (p. 33) has.*
- unsigned **getLasersOnScreen** () const

  Check how many lasers there are on screen.
- unsigned **getBonusesOnScreen** () const

  Check how many bonuses there are on screen.
- void **reset** (const unsigned x, const unsigned y)

  *Reset **Invader** (p. 33).*
- void **die** ()

  *Kill **Invader** (p. 33).*
- void **move** ()

  *Move **Invader** (p. 33).*
- void **dropDown** ()

  The invader lowers by one row.

- void **reverseDir** ()

  The invader reverses its direction of movement.
- bool **checkHitEdge** (const int screenw)

  *Did **Invader** (p. 33) hit edge of screen?*
- void **incDeathTick** ()

  Increment death tick and hide once hit max.
- void **incLasersOnScreen** ()

  Set one more laser on screen.
- void **decLasersOnScreen** ()

  Erase one laser from screen.
- void **incBonusesOnScreen** ()

  Set one more bonus on screen.
- void **decBonusesOnScreen** ()

  Erase one bonus from screen.

Static Public Member Functions

- static unsigned **getHeight** ()

  Get how much invader should drop down.

5.6.1 Detailed Description

Class for **Invader** (p. 33).

Name: **Invader.h** (p. 98) Purpose: Declaration of class **Invader** (p. 33)

Author

Fenris

Version

0.94b 03/05/2017

5.6.2 Member Enumeration Documentation

5.6.2.1 InvaderType

`enum` **Invader::InvaderType**

Container for types of Invaders.

Enumerator

INVADER_1	
INVADER_2	
INVADER_3	
INVADER_	

5.6.3 Constructor & Destructor Documentation

5.6.3.1 Invader()

```
Invader::Invader (
              Textures & textures,
             const  InvaderType type,
             int isVisible )
```

Default constructor for **Invader** (p. 33).

Parameters

textures	Texture for **Invader** (p. 33)'s sprite
type	Type of **Invader** (p. 33)
isVisible	Is **Invader** (p. 33) visible?

All Invaders have the same death

5.6.4 Member Function Documentation

5.6.4.1 checkHitEdge()

```
bool Invader::checkHitEdge (
             const int screenw )
```

Did **Invader** (p. 33) hit edge of screen?

Parameters

screenw	Width of screen

5.6.4.2 decBonusesOnScreen()

```
void Invader::decBonusesOnScreen ( )  [inline]
```

Erase one bonus from screen.

5.6.4.3 decLasersOnScreen()

```
void Invader::decLasersOnScreen ( )  [inline]
```

Erase one laser from screen.

5.6.4.4 die()

```
void Invader::die ( )
```

Kill **Invader** (p. 33).

5.6.4.5 dropDown()

```
void Invader::dropDown ( )
```

The invader lowers by one row.

5.6.4.6 getBonusesOnScreen()

```
unsigned Invader::getBonusesOnScreen ( ) const  [inline]
```

Check how many bonuses there are on screen.

5.6.4.7 getHeight()

```
static unsigned Invader::getHeight ( )  [inline], [static]
```

Get how much invader should drop down.

5.6.4.8 getLasersOnScreen()

```
unsigned Invader::getLasersOnScreen ( ) const  [inline]
```

Check how many lasers there are on screen.

5.6.4.9 getLives()

```
int Invader::getLives ( ) const  [inline]
```

Check how many lives **Invader** (p. 33) has.

5.6.4.10 getMoveDir()

```
int Invader::getMoveDir ( ) const  [inline]
```

Check movement direction.

5.6.4.11 getScoreValue()

```
unsigned Invader::getScoreValue ( ) const  [inline]
```

Get how mush score user will get.

5.6.4.12 getSprite()

```
sf::Sprite& Invader::getSprite ( )  [inline]
```

Get **Invader** (p. 33)'s sprite.

5.6.4.13 incBonusesOnScreen()

```
void Invader::incBonusesOnScreen ( )  [inline]
```

Set one more bonus on screen.

5.6.4.14 incDeathTick()

```
void Invader::incDeathTick ( )
```

Increment death tick and hide once hit max.

5.6.4.15 incLasersOnScreen()

```
void Invader::incLasersOnScreen ( )  [inline]
```

Set one more laser on screen.

5.6.4.16 isDead()

```
bool Invader::isDead ( ) const  [inline]
```

Check if **Invader** (p. 33) is dead.

5.6.4.17 isExploding()

```
bool Invader::isExploding ( ) const  [inline]
```

Check if **Invader** (p. 33) is exploding.

5.6.4.18 isVisible()

```
bool Invader::isVisible ( ) const  [inline]
```

Check if **Invader** (p. 33) is visible.

5.6.4.19 move()

```
void Invader::move ( )
```

Move **Invader** (p. 33).

5.6.4.20 reset()

```
void Invader::reset (
            const unsigned x,
            const unsigned y )
```

Reset **Invader** (p. 33).

Parameters

x	X Position for **Invader** (p. 33)
y	Y Position for **Invader** (p. 33)

Name: **Invader.cpp** (p. 98) Purpose: Class **Invader** (p. 33)

Author

Fenris

Version

0.94a 03/05/2017

5.6.4.21 reverseDir()

```
void Invader::reverseDir ( )
```

The invader reverses its direction of movement.

5.6.4.22 setLives()

```
void Invader::setLives (
            int life )  [inline]
```

Set how many lives should **Invader** (p. 33) has.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Invader.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Invader.cpp**

5.7 InvaderFormation Class Reference

Class of **InvaderFormation** (p. 39).

```
#include <InvaderFormation.h>
```

Public Member Functions

- **InvaderFormation** (sf::RenderWindow &window)
  *Default constructor for **InvaderFormation** (p. 39).*
- ~**InvaderFormation** ()
  *Default destructor for **InvaderFormation** (p. 39).*
- **Lasers** & **getLasers** ()
  Get vector with lasers.
- **Bonuses** & **getBonuses** ()
  Get vector with bonuses.
- **InvaderVector2D** & **getInvaders** ()
  Get vector with formation.
- unsigned **getTotal** () const
  Get how many invaders there are to kill.
- unsigned **getNumKilled** () const
  Get how many invaders there are already killed.
- void **reset** ()
  Reset the formation.
- void **update** (**PlayerLaser** &laser, **Spaceship** &ship, **PlayerLaser** &player_laser, **LivesDisplay** &lives_↩ disp, **Explosions** &explosions, unsigned &game_score, unsigned &minutes, unsigned &seconds, unsigned &miliseconds)
  Update the formation and it's movement.
- void **draw** (int amount=-1)
  Draw formation.
- void **drawLasers** ()
  Draw lasers.
- void **drawBonuses** ()
  Draw bonuses.
- void **clearLevel** ()
  Clear level from invisible invaders.
- void **loadLevel** (sf::RenderWindow &window, **Textures** &textures, std::string fileName)
  ***Load** (p. 51) level from files.*
- void **removeHitBonuses** ()
  Remove bonuses that got hit.
- void **removeLasers** ()
  Remove all lasers.
- void **removeBonuses** ()
  Remove all bonuses.

5.7.1 Detailed Description

Class of **InvaderFormation** (p. 39).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 InvaderFormation()

```
InvaderFormation::InvaderFormation (
            sf::RenderWindow & window )
```

Default constructor for **InvaderFormation** (p. 39).

Parameters

window	Where it should be displayed

5.7.2.2 ~InvaderFormation()

```
InvaderFormation::~InvaderFormation ( )
```

Default destructor for **InvaderFormation** (p. 39).

5.7.3 Member Function Documentation

5.7.3.1 clearLevel()

```
void InvaderFormation::clearLevel ( )
```

Clear level from invisible invaders.

5.7.3.2 draw()

```
void InvaderFormation::draw (
            int amount = -1 )
```

Draw formation.

Parameters

amount	

5.7.3.3 drawBonuses()

```
void InvaderFormation::drawBonuses ( )
```

Draw bonuses.

5.7.3.4 drawLasers()

```
void InvaderFormation::drawLasers ( )
```

Draw lasers.

5.7.3.5 getBonuses()

```
Bonuses& InvaderFormation::getBonuses ( )  [inline]
```

Get vector with bonuses.

Returns

Vector with bonuses

5.7.3.6 getInvaders()

```
InvaderVector2D& InvaderFormation::getInvaders ( )  [inline]
```

Get vector with formation.

Returns

Vector with InvaderRows

5.7.3.7 getLasers()

```
Lasers& InvaderFormation::getLasers ( )  [inline]
```

Get vector with lasers.

Returns

Vector with lasers

5.7.3.8 getNumKilled()

```
unsigned InvaderFormation::getNumKilled ( ) const  [inline]
```

Get how many invaders there are already killed.

5.7.3.9 getTotal()

```
unsigned InvaderFormation::getTotal ( ) const  [inline]
```

Get how many invaders there are to kill.

5.7.3.10 loadLevel()

```
void InvaderFormation::loadLevel (
            sf::RenderWindow & window,
             Textures & textures,
            std::string fileName )
```

Load (p. 51) level from files.

Vector for each row in the formation

Now add each row to the main vector

5.7.3.11 removeBonuses()

```
void InvaderFormation::removeBonuses ( )
```

Remove all bonuses.

5.7.3.12 removeHitBonuses()

```
void InvaderFormation::removeHitBonuses ( )
```

Remove bonuses that got hit.

5.7.3.13 removeLasers()

```
void InvaderFormation::removeLasers ( )
```

Remove all lasers.

5.7.3.14 reset()

```
void InvaderFormation::reset ( )
```

Reset the formation.

5.7.3.15 update()

```
void InvaderFormation::update (
              PlayerLaser & laser,
              Spaceship & ship,
              PlayerLaser & player_laser,
              LivesDisplay & lives_disp,
              Explosions & explosions,
             unsigned & game_score,
             unsigned & minutes,
             unsigned & seconds,
             unsigned & miliseconds )
```

Update the formation and it's movement.

Parameters

laser	For updating movement
ship	For updating movement
player_laser	For updating lasers
lives_disp	For checking if there was a hit
explosions	For updating lasers
game_score	For updating score
minutes	For updating clock
seconds	For updating clock
miliseconds	For updating clock

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **InvaderFormation.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **InvaderFormation.cpp**

5.8 InvaderLaser Class Reference

Class for **InvaderLaser** (p. 44).

```
#include <InvaderLaser.h>
```

Public Member Functions

- **InvaderLaser** (const unsigned x, const unsigned y, const bool will_hurt, **Invader** &owner)

  *Default constructor for **InvaderLaser** (p. 44).*
- ∼**InvaderLaser** ()

  *Default destructor for **InvaderLaser** (p. 44).*
- bool **isHit** () const

  Check if the laser was hit.
- bool **willHurt** () const

  Check if it will hurt.
- void **setHit** ()

Set that laser was hit.

- unsigned **getX** () const

  Check the X Position of the laser.

- unsigned **getY** () const

  Check the Y Position of the laser.

- void **move** ()

  Move laser.

- void **draw** (sf::RenderWindow &window)

  Draw laser on the scene.

- bool **checkCollide** (const unsigned x, const unsigned y) const

  Check collisions with players ship.

- bool **checkCollide** (const sf::FloatRect rect) const

  Check collisions with players ship.

5.8.1 Detailed Description

Class for **InvaderLaser** (p. 44).

Name: **InvaderLaser.h** (p. 100) Purpose: Declaration of class **InvaderLaser** (p. 44)

Author

Fenris

Version

0.81b 03/05/2017

5.8.2 Constructor & Destructor Documentation

5.8.2.1 InvaderLaser()

```
InvaderLaser::InvaderLaser (
            const unsigned x,
            const unsigned y,
            const bool will_hurt,
             Invader & owner )
```

Default constructor for **InvaderLaser** (p. 44).

Parameters

x	X Position of the laser
y	Y Position of the laser
will_hurt	Whether or not the laser will hurt the player
owner	Whose laser is this?

Name: **InvaderLaser.cpp** (p. 100) Purpose: Class **InvaderLaser** (p. 44)

Author

Fenris

Version

0.88a 03/05/2017

5.8.2.2 ~InvaderLaser()

```
InvaderLaser::~InvaderLaser ( )  [inline]
```

Default destructor for **InvaderLaser** (p. 44).

5.8.3 Member Function Documentation

5.8.3.1 checkCollide() [1/2]

```
bool InvaderLaser::checkCollide (
            const unsigned x,
            const unsigned y ) const
```

Check collisions with players ship.

Parameters

x	X Position of players ship
y	Y Position of players ship

5.8.3.2 checkCollide() [2/2]

```
bool InvaderLaser::checkCollide (
            const sf::FloatRect rect ) const
```

Check collisions with players ship.

Parameters

rect	Rectangle of players ship

5.8.3.3 draw()

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

Draw laser on the scene.

Parameters

window	Where it will be displayed?

5.8.3.4 getX()

```
unsigned InvaderLaser::getX ( ) const
```

Check the X Position of the laser.

5.8.3.5 getY()

```
unsigned InvaderLaser::getY ( ) const
```

Check the Y Position of the laser.

5.8.3.6 isHit()

```
bool InvaderLaser::isHit ( ) const  [inline]
```

Check if the laser was hit.

5.8.3.7 move()

```
void InvaderLaser::move ( )
```

Move laser.

5.8.3.8 setHit()

```
void InvaderLaser::setHit ( )  [inline]
```

Set that laser was hit.

5.8.3.9 willHurt()

```
bool InvaderLaser::willHurt ( ) const  [inline]
```

Check if it will hurt.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **InvaderLaser.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **InvaderLaser.cpp**

5.9 LivesDisplay Class Reference

Class for Lives Display.

```
#include <LivesDisplay.h>
```

Public Member Functions

- **LivesDisplay** (**Textures** &textures)

  *Default constructor for **LivesDisplay*** (p. *48*).
- ∼**LivesDisplay** ()

  *Default destructor for **LivesDisplay*** (p. *48*).
- unsigned **getLives** () const

  How many lives there is?
- void **reset** ()

  Reset display to show 0 lives.
- void **removeLife** ()

  Lose one life.
- void **addLife** ()

  Add one life.
- void **setLives** (const unsigned num)

  Set how many lives there should be.
- void **draw** (sf::RenderWindow &window)

  *Draw **LivesDisplay*** (p. *48*) *on scene.*

5.9.1 Detailed Description

Class for Lives Display.

Name: **LivesDisplay.h** (p. 100) Purpose: Declaration of class **LivesDisplay** (p. 48)

Author

Fenris

Version

0.86a 02/05/2017

5.9.2 Constructor & Destructor Documentation

5.9.2.1 LivesDisplay()

```
LivesDisplay::LivesDisplay (
              Textures & textures )
```

Default constructor for **LivesDisplay** (p. 48).

Parameters

textures	**Textures** (p. 78) for lives to display

5.9.2.2 ~LivesDisplay()

```
LivesDisplay::~LivesDisplay ( )
```

Default destructor for **LivesDisplay** (p. 48).

5.9.3 Member Function Documentation

5.9.3.1 addLife()

```
void LivesDisplay::addLife ( )
```

Add one life.

5.9.3.2 draw()

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

Draw **LivesDisplay** (p. 48) on scene.

Parameters

window	Where it should be displayed

5.9.3.3 getLives()

```
unsigned LivesDisplay::getLives ( ) const  [inline]
```

How many lives there is?

5.9.3.4 removeLife()

```
void LivesDisplay::removeLife ( )
```

Lose one life.

5.9.3.5 reset()

```
void LivesDisplay::reset ( )
```

Reset display to show 0 lives.

Name: **LivesDisplay.cpp** (p. 100) Purpose: Class **LivesDisplay** (p. 48)

Author

Fenris

Version

0.94a 03/05/2017

5.9.3.6 setLives()

```
void LivesDisplay::setLives (
            const unsigned num )
```

Set how many lives there should be.

Parameters

num	Number of lives to set

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **LivesDisplay.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **LivesDisplay.cpp**

5.10 Load Class Reference

Class of State LOAD.

`#include <Load.h>`

Public Member Functions

- **Load** (**Textures** &textures, **ScoreDisplay** &score_disp, **ClockDisplay** &clock_disp, **LivesDisplay** &lives, unsigned &wave_on, unsigned &score, unsigned &minutes, unsigned &seconds, unsigned &miliseconds)

  *Default constructor for **Load** (p. 51).*
- void **reset** ()

  *Reset **Load** (p. 51) to get specific type of drawing.*
- void **doLoad** ()

  ***Load** (p. 51) variables from file.*
- void **drawLine** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, const unsigned wait, const unsigned line_num, sf::Sprite ∗sprite=nullptr, sf::Color color=sf::Color::White, unsigned size=24)

  Draws a string with pauses after each character (wait)
- void **draw** (sf::RenderWindow &window)

  *Draw **Load** (p. 51).*
- void **update** (**Textures** &textures)

  Update arrow to get new frame.

Public Attributes

- std::string **fileName**

  Name of file to load.

5.10.1 Detailed Description

Class of State LOAD.

Name: **Load.h** (p. 101) Purpose: Declaration of class **Load** (p. 51)

Author

Fenris

Version

1.04 14/05/2017

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Load()

```
Load::Load (
              Textures & textures,
              ScoreDisplay & score_disp,
              ClockDisplay & clock_disp,
              LivesDisplay & lives,
             unsigned & wave_on,
             unsigned & score,
             unsigned & minutes,
             unsigned & seconds,
             unsigned & miliseconds )
```

Default constructor for **Load** (p. 51).

Parameters

textures	**Textures** (p. 78) which will be displayed next to score
score_disp	Score displayed on scene
clock_disp	Clock displayed on scene
lives	Lives displayed on scene
wave_on	Which level is it?
score	How many points you have?
minutes	Minutes which were read from clock
seconds	Seconds which were read from clock
miliseconds	Miliseconds which were read from clock

Name: **Load.cpp** (p. 101) Purpose: Class **Load** (p. 51)

Author

Fenris

Version

1.08a 14/05/2017

5.10.3 Member Function Documentation

5.10.3.1 doLoad()

```
void Load::doLoad ( )
```

Load (p. 51) variables from file.

5.10.3.2 draw()

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

Draw **Load** (p. 51).

Parameters

window	Where should it be displayed

5.10.3.3 drawLine()

```
void Load::drawLine (
            sf::RenderWindow & window,
            const std::string msg,
            const unsigned x,
            const unsigned y,
            const unsigned wait,
            const unsigned line_num,
            sf::Sprite * sprite = nullptr,
            sf::Color color = sf::Color::White,
            unsigned size = 24 )
```

Draws a string with pauses after each character (wait)

Parameters

window	Where should it be displayed
msg	What message should be displayed
x	X Position where it should be displayed on scene
y	Y Position where it should be displayed on scene
wait	How much time pause should take
line_num	What line is it on?
sprite	What sprite should be displayed next to text
color	In which color text should be displayed?
size	What size of font should text has?

Don't do anything until **Load** (p. 51) isn't done with previous line

If it's okey it will drew next line

Draw the line letter by letter

5.10.3.4 reset()

```
void Load::reset ( )
```

Reset **Load** (p. 51) to get specific type of drawing.

5.10.3.5 update()

```
void Load::update (
            Textures & textures )
```

Update arrow to get new frame.

Parameters

textures	**Textures** (p. 78) for animated arrow

5.10.4 Member Data Documentation

5.10.4.1 fileName

```
std::string Load::fileName
```

Name of file to load.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Load.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Load.cpp**

5.11 Menu Class Reference

Class of State MENU.

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

Public Member Functions

- **Menu** (**Textures** &textures, **ScoreDisplay** &score_disp, **ClockDisplay** &clock_disp)

  *Default constructor for **Info** (p. 30).*
- void **setSelect** (int select)

  Set which option is selected.
- int **getSelect** ()

  Get which option is selected.
- void **reset** ()

  *Reset **Menu** (p. 54) to get specific type of drawing.*
- void **drawLine** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, const unsigned wait, const unsigned line_num, sf::Sprite ∗sprite=nullptr, sf::Color color=sf::Color::White, unsigned size=24)

  Draws a string with pauses after each character (wait)
- void **draw** (sf::RenderWindow &window)

  *Draw **Menu** (p. 54).*
- void **update** (**Textures** &textures)

  Update arrow to get new frame.

5.11.1 Detailed Description

Class of State MENU.

Name: **Menu.h** (p. 102) Purpose: Declaration of class **Menu** (p. 54)

Author

Fenris

Version

1.04 14/05/2017

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Menu()

```
Menu::Menu (
              Textures & textures,
              ScoreDisplay & score_disp,
              ClockDisplay & clock_disp )
```

Default constructor for **Info** (p. 30).

Parameters

textures	**Textures** (p. 78) which will be displayed next to score
score_disp	Score displayed on scene
clock_disp	Clock displayed on scene

Name: **Menu.cpp** (p. 102) Purpose: Class **Menu** (p. 54)

Author

Fenris

Version

1.08a 14/05/2017

5.11.3 Member Function Documentation

5.11.3.1 draw()

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

Draw **Menu** (p. 54).

Parameters

window	Where should it be displayed

5.11.3.2 drawLine()

```
void Menu::drawLine (
            sf::RenderWindow & window,
            const std::string msg,
            const unsigned x,
            const unsigned y,
            const unsigned wait,
            const unsigned line_num,
            sf::Sprite * sprite = nullptr,
            sf::Color color = sf::Color::White,
            unsigned size = 24 )
```

Draws a string with pauses after each character (wait)

Parameters

window	Where should it be displayed
msg	What message should be displayed
x	X Position where it should be displayed on scene
y	Y Position where it should be displayed on scene
wait	How much time pause should take
line_num	What line is it on?
sprite	What sprite should be displayed next to text
color	In which color text should be displayed?
size	What size of font should text has?

Don't do anything until menu isn't done with previous line

If it's okey it will drew next line

Draw the line letter by letter

5.11.3.3 getSelect()

```
int Menu::getSelect ( )  [inline]
```

Get which option is selected.

5.11.3.4 reset()

```
void Menu::reset ( )
```

Reset **Menu** (p. 54) to get specific type of drawing.

5.11.3.5 setSelect()

```
void Menu::setSelect (
             int select )  [inline]
```

Set which option is selected.

Parameters

select	Option which will be selected

5.11.3.6 update()

```
void Menu::update (
              Textures & textures )
```

Update arrow to get new frame.

Parameters

textures	**Textures** (p. 78) for animated arrow

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Menu.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Menu.cpp**

5.12 MenuIG Class Reference

Class of State MENUIG.

```
#include <Menu_in_game.h>
```

Public Member Functions

- **MenuIG** (**Textures** &textures, **ScoreDisplay** &score_disp, **ClockDisplay** &clock_disp)

  *Default constructor for **Info** (p. 30).*
- void **setSelect** (int select)

  Set which option is selected.
- int **getSelect** ()

  Get which option is selected.
- void **reset** ()

  *Reset **Menu** (p. 54) In **Game** (p. 7) to get specific type of drawing.*
- void **drawLine** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, const unsigned wait, const unsigned line_num, sf::Sprite ∗sprite=nullptr, sf::Color color=sf::Color::White, unsigned size=24)

  Draws a string with pauses after each character (wait)
- void **draw** (sf::RenderWindow &window)

  *Draw **Menu** (p. 54) In **Game** (p. 7).*
- void **update** (**Textures** &textures)

  Update arrow to get new frame.

5.12.1 Detailed Description

Class of State MENUIG.

Name: **Menu_in_game.h** (p. 103) Purpose: Declaration of class **MenuIG** (p. 57)

Author

Fenris

Version

1.04 14/05/2017

5.12.2 Constructor & Destructor Documentation

5.12.2.1 MenuIG()

```
MenuIG::MenuIG (
            Textures & textures,
            ScoreDisplay & score_disp,
            ClockDisplay & clock_disp )
```

Default constructor for **Info** (p. 30).

Parameters

textures	**Textures** (p. 78) which will be displayed next to score
score_disp	Score displayed on scene
clock_disp	Clock displayed on scene

Name: **Menu_in_game.cpp** (p. 102) Purpose: Class **MenuIG** (p. 57)

Author

Fenris

Version

1.08a 14/05/2017

5.12.3 Member Function Documentation

5.12.3.1 draw()

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

Draw **Menu** (p. 54) In **Game** (p. 7).

Parameters

window	Where should it be displayed

5.12.3.2 drawLine()

```
void MenuIG::drawLine (
            sf::RenderWindow & window,
            const std::string msg,
            const unsigned x,
            const unsigned y,
            const unsigned wait,
            const unsigned line_num,
            sf::Sprite * sprite = nullptr,
            sf::Color color = sf::Color::White,
            unsigned size = 24 )
```

Draws a string with pauses after each character (wait)

Parameters

window	Where should it be displayed
msg	What message should be displayed
x	X Position where it should be displayed on scene
y	Y Position where it should be displayed on scene
wait	How much time pause should take

Parameters

line_num	What line is it on?
sprite	What sprite should be displayed next to text
color	In which color text should be displayed?
size	What size of font should text has?

Don't do anything until **MenuIG** (p. 57) isn't done with previous line

If it's okey it will drew next line

Draw the line letter by letter

5.12.3.3 getSelect()

```
int MenuIG::getSelect ( )  [inline]
```

Get which option is selected.

5.12.3.4 reset()

```
void MenuIG::reset ( )
```

Reset **Menu** (p. 54) In **Game** (p. 7) to get specific type of drawing.

5.12.3.5 setSelect()

```
void MenuIG::setSelect (
            int select )  [inline]
```

Set which option is selected.

Parameters

select	Option which will be selected

5.12.3.6 update()

```
void MenuIG::update (
             Textures & textures )
```

Update arrow to get new frame.

Parameters

textures	**Textures** (p. 78) for animated arrow

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Menu_in_game.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Menu_in_game.cpp**

5.13 PlayerLaser Class Reference

Class for **PlayerLaser** (p. 61).

`#include <PlayerLaser.h>`

Public Member Functions

- **PlayerLaser** ()
  Default constructor of Player laser.
- sf::RectangleShape & **getShape** ()
  Get shape of first laser.
- sf::RectangleShape & **getShape2** ()
  Get shape of second laser.
- bool **isShooting** () const
  Check the first laser (Is it shooting?)
- bool **isShooting2** () const
  Check the second laser (Is it shooting?)
- bool **isDouble** () const
  Check if there is a bonus.
- void **nowDouble** ()
  Get the bonus.
- int **getSpeed** ()
  Check the speed of the laser.
- void **setSpeed** (int speed)
  Set the speed of the laser.
- bool **getStop1** ()
  Check if the first laser stopped shooting.
- bool **getStop2** ()
  Check if the second laser stopped shooting.
- void **shoot** (const unsigned startx, const unsigned starty)
  Shoot the laser.
- void **move** (**Explosions** &explosions)
  Move the laser.
- void **stop1** ()
  Stop the first laser.
- void **stop2** ()
  Stop the second laser.
- void **reset** ()
  Reset player laser (Laser won't be double and speed will has the starting value)

5.13.1 Detailed Description

Class for **PlayerLaser** (p. 61).

Name: **PlayerLaser.h** (p. 103) Purpose: Declaration of class **PlayerLaser** (p. 61)

Author

Fenris

Version

0.88a 03/05/2017

5.13.2 Constructor & Destructor Documentation

5.13.2.1 PlayerLaser()

```
PlayerLaser::PlayerLaser ( )
```

Default constructor of Player laser.

Name: **PlayerLaser.cpp** (p. 103) Purpose: Class **PlayerLaser** (p. 61)

Author

Fenris

Version

0.94a 03/05/2017

5.13.3 Member Function Documentation

5.13.3.1 getShape()

```
sf::RectangleShape& PlayerLaser::getShape ( )  [inline]
```

Get shape of first laser.

Returns

The shape of laser

5.13.3.2 getShape2()

```
sf::RectangleShape& PlayerLaser::getShape2 ( )  [inline]
```

Get shape of second laser.

Returns

The shape of laser

5.13.3.3 getSpeed()

```
int PlayerLaser::getSpeed ( )  [inline]
```

Check the speed of the laser.

5.13.3.4 getStop1()

```
bool PlayerLaser::getStop1 ( )  [inline]
```

Check if the first laser stopped shooting.

5.13.3.5 getStop2()

```
bool PlayerLaser::getStop2 ( )  [inline]
```

Check if the second laser stopped shooting.

5.13.3.6 isDouble()

```
bool PlayerLaser::isDouble ( ) const  [inline]
```

Check if there is a bonus.

5.13.3.7 isShooting()

```
bool PlayerLaser::isShooting ( ) const  [inline]
```

Check the first laser (Is it shooting?)

5.13.3.8 isShooting2()

```
bool PlayerLaser::isShooting2 ( ) const  [inline]
```

Check the second laser (Is it shooting?)

5.13.3.9 move()

```
void PlayerLaser::move (
            Explosions & explosions )
```

Move the laser.

Parameters

explosions	**Explosions** (p. 28) after collision with something

5.13.3.10 nowDouble()

```
void PlayerLaser::nowDouble ( )  [inline]
```

Get the bonus.

5.13.3.11 reset()

```
void PlayerLaser::reset ( )
```

Reset player laser (Laser won't be double and speed will has the starting value)

5.13.3.12 setSpeed()

```
void PlayerLaser::setSpeed (
            int speed )  [inline]
```

Set the speed of the laser.

Parameters

speed	Value of speed that will be set

5.13.3.13 shoot()

```
void PlayerLaser::shoot (
            const unsigned startx,
            const unsigned starty )
```

Shoot the laser.

Parameters

startx	X Position of the laser that will be displayed
starty	Y Position of the laser that will be displayed

You can't shoot while it already has been shot

5.13.3.14 stop1()

```
void PlayerLaser::stop1 ( )
```

Stop the first laser.

5.13.3.15 stop2()

```
void PlayerLaser::stop2 ( )
```

Stop the second laser.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **PlayerLaser.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **PlayerLaser.cpp**

5.14 Save Class Reference

Class of State SAVE.

```
#include <Save.h>
```

Public Member Functions

- **Save** (**Textures** &textures, **ScoreDisplay** &score_disp, **ClockDisplay** &clock_disp, **LivesDisplay** &lives, unsigned &wave_on, unsigned &score, unsigned &minutes, unsigned &seconds, unsigned &miliseconds)

  *Default constructor for **Save** (p. 65).*
- void **reset** ()

  *Reset **Save** (p. 65) to get specific type of drawing.*
- void **doSave** ()

  ***Save** (p. 65) variables to file.*
- void **drawLine** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, const unsigned wait, const unsigned line_num, sf::Sprite *sprite=nullptr, sf::Color color=sf::Color::White, unsigned size=24)

  Draws a string with pauses after each character (wait)
- void **draw** (sf::RenderWindow &window)

  *Draw **Save** (p. 65).*
- void **update** (**Textures** &textures)

  Update arrow to get new frame.

Public Attributes

- std::string **fileName**

  Name of file for saving.

5.14.1 Detailed Description

Class of State SAVE.

Name: **Save.h** (p. 104) Purpose: Declaration of class **Save** (p. 65)

Author

Fenris

Version

1.04 14/05/2017

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Save()

```
Save::Save (
             Textures & textures,
             ScoreDisplay & score_disp,
             ClockDisplay & clock_disp,
             LivesDisplay & lives,
            unsigned & wave_on,
            unsigned & score,
            unsigned & minutes,
            unsigned & seconds,
            unsigned & miliseconds )
```

Default constructor for **Save** (p. 65).

Parameters

textures	**Textures** (p. 78) which will be displayed next to score
score_disp	Score displayed on scene
clock_disp	Clock displayed on scene
lives	Lives displayed on scene
wave_on	Which level is it?
score	How many points you have?
minutes	Minutes which were read from clock
seconds	Seconds which were read from clock
miliseconds	Miliseconds which were read from clock

Name: **Save.cpp** (p. 104) Purpose: Class **Save** (p. 65)

Author

Fenris

Version

1.08a 14/05/2017

5.14.3 Member Function Documentation

5.14.3.1 doSave()

```
void Save::doSave ( )
```

Save (p. 65) variables to file.

5.14.3.2 draw()

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

Draw **Save** (p. 65).

Parameters

window	Where should it be displayed

5.14.3.3 drawLine()

```
void Save::drawLine (
            sf::RenderWindow & window,
            const std::string msg,
            const unsigned x,
            const unsigned y,
            const unsigned wait,
            const unsigned line_num,
            sf::Sprite * sprite = nullptr,
            sf::Color color = sf::Color::White,
            unsigned size = 24 )
```

Draws a string with pauses after each character (wait)

Parameters

window	Where should it be displayed
msg	What message should be displayed
x	X Position where it should be displayed on scene
y	Y Position where it should be displayed on scene
wait	How much time pause should take
line_num	What line is it on?
sprite	What sprite should be displayed next to text
color	In which color text should be displayed?
size	What size of font should text has?

Don't do anything until **Save** (p. 65) isn't done with previous line

If it's okey it will drew next line

Draw the line letter by letter

5.14.3.4 reset()

```
void Save::reset ( )
```

Reset **Save** (p. 65) to get specific type of drawing.

5.14.3.5 update()

```
void Save::update (
            Textures & textures )
```

Update arrow to get new frame.

Parameters

textures	**Textures** (p. 78) for animated arrow

5.14.4 Member Data Documentation

5.14.4.1 fileName

```
std::string Save::fileName
```

Name of file for saving.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Save.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Save.cpp**

5.15 ScoreDisplay Class Reference

Class for Score Display.

```
#include <ScoreDisplay.h>
```

Public Member Functions

- **ScoreDisplay** (unsigned &score)
  *Default constructor for **ScoreDisplay** (p. 69).*
- void **drawScore** (sf::RenderWindow &window, const unsigned score, const unsigned x, const unsigned y)
  Get text to display on scene.
- void **draw** (sf::RenderWindow &window)
  Draw Score(text) on scene.
- void **reset** ()
  Reset score to display 0.

5.15.1 Detailed Description

Class for Score Display.

Name: **ScoreDisplay.h** (p. 104) Purpose: Declaration of class **ScoreDisplay** (p. 69)

Author

Fenris

Version

0.76 03/05/2017

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ScoreDisplay()

```
ScoreDisplay::ScoreDisplay (
            unsigned & score )
```

Default constructor for **ScoreDisplay** (p. 69).

Parameters

score	Score which will be displayed

Name: **ScoreDisplay.cpp** (p. 104) Purpose: Class **ScoreDisplay** (p. 69)

Author

Fenris

Version

0.84a 03/05/2017

5.15.3 Member Function Documentation

5.15.3.1 draw()

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

Draw Score(text) on scene.

5.15.3.2 drawScore()

```
void ScoreDisplay::drawScore (
            sf::RenderWindow & window,
            const unsigned score,
            const unsigned x,
            const unsigned y )
```

Get text to display on scene.

Parameters

window	Where it should be drawn
score	Score to be displayed
x	X Position of **ScoreDisplay** (p. 69) on scene
y	Y Position of **ScoreDisplay** (p. 69) on scene

5.15.3.3 reset()

```
void ScoreDisplay::reset ( )
```

Reset score to display 0.

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **ScoreDisplay.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **ScoreDisplay.cpp**

5.16 Spaceship Class Reference

Class for **Spaceship** (p. 71).

```
#include <Spaceship.h>
```

Public Member Functions

- **Spaceship** (**Textures** &textures, const int startx)
  *Default constructor of **Spaceship** (p. 71).*
- bool **isHit** () const
  Check if it was hit.
- void **reset** ()
  Reset the spaceship.
- void **move** (const int dir)
  Move the ship.
- const sf::Sprite & **getSprite** () const
  Get the sprite of the ship.
- unsigned **getX** () const
  Check the X Position of the ship.
- unsigned **getWidth** () const
  Check the width of the sprite.
- void **die** (**InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Explosions** &explosions)
  Die.
- void **handleHit** (**InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Explosions** &explosions, **LivesDisplay** &lives, unsigned &game_score)
  Handle hit from bonuses and lasers.
- void **update** (**InvaderFormation** &invaders, **PlayerLaser** &player_laser, **LivesDisplay** &lives_disp, **Explosions** &explosions, unsigned &game_score)
  Update ship while playing.

5.16.1 Detailed Description

Class for **Spaceship** (p. 71).

Name: **Spaceship.h** (p. 105) Purpose: Declaration of class **Spaceship** (p. 71)

Author

Fenris

Version

0.84a 03/05/2017

5.16.2 Constructor & Destructor Documentation

5.16.2.1 Spaceship()

```
Spaceship::Spaceship (
              Textures & textures,
             const int startx )
```

Default constructor of **Spaceship** (p. 71).

Parameters

textures	**Textures** (p. 78) that will be displayed on scene
startx	X Position of the ship while starting the game

5.16.3 Member Function Documentation

5.16.3.1 die()

```
void Spaceship::die (
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              Explosions & explosions )
```

Die.

Parameters

invaders	For removal of theirs lasers
player_laser	To stop all players lasers
explosions	To clear all explosions

5.16.3.2 getSprite()

```
const sf::Sprite& Spaceship::getSprite ( ) const  [inline]
```

Get the sprite of the ship.

Returns

Sprite of the ship with all parameters

5.16.3.3 getWidth()

```
unsigned Spaceship::getWidth ( ) const  [inline]
```

Check the width of the sprite.

5.16.3.4 getX()

```
unsigned Spaceship::getX ( ) const  [inline]
```

Check the X Position of the ship.

5.16.3.5 handleHit()

```
void Spaceship::handleHit (
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              Explosions & explosions,
              LivesDisplay & lives,
             unsigned & game_score )
```

Handle hit from bonuses and lasers.

Parameters

invaders	For removal of theirs lasers
player_laser	To stop all player lasers and to speed up or decelerate them
explosions	To remove all explosions
lives	To change lives
game_score	To change the game score

5.16.3.6 isHit()

```
bool Spaceship::isHit ( ) const  [inline]
```

Check if it was hit.

5.16.3.7 move()

```
void Spaceship::move (
             const int dir )
```

Move the ship.

Parameters

dir	Direction which depends on key that we pressed on keyboard

5.16.3.8 reset()

```
void Spaceship::reset ( )
```

Reset the spaceship.

5.16.3.9 update()

```
void Spaceship::update (
              InvaderFormation & invaders,
              PlayerLaser & player_laser,
              LivesDisplay & lives_disp,
              Explosions & explosions,
             unsigned & game_score )
```

Update ship while playing.

Parameters

invaders	For removal of theirs lasers
player_laser	To stop all player lasers and to speed up or decelerate them
explosions	To remove all explosions
lives_disp	To change lives
game_score	To change the game score

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Spaceship.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Spaceship.cpp**

5.17 Stars Class Reference

Class of **Stars** (p. 74).

```
#include <Stars.h>
```

Public Member Functions

- **Stars** (sf::RenderWindow &window, const unsigned x, float scale)
  *Default constructor of **Stars** (p. 74).*
- ∼**Stars** ()
  *Default destructor of **Stars** (p. 74).*
- bool **isHit** () const
  See if star was hit.
- void **setHit** ()
  Set that this star got hit.
- **Constellation** & **getStars** ()
  Get the constellation.
- void **updateStars** (sf::RenderWindow &window)
  Update stars to get them moving and shooting.
- void **removeHitStars** ()
  Remove stars that got hit.
- void **drawStars** ()
  Draw stars.
- void **checkHitEdge** ()
  Check if star gone to far.
- unsigned **getX** () const
  Get X Position of star.
- unsigned **getY** () const
  Get Y Position of star.
- void **move** ()
  Move star.
- void **draw** (sf::RenderWindow &window)
  Display stars on scene.

5.17.1 Detailed Description

Class of **Stars** (p. 74).

5.17.2 Constructor & Destructor Documentation

5.17.2.1 Stars()

```
Stars::Stars (
            sf::RenderWindow & window,
            const unsigned x,
            float scale )
```

Default constructor of **Stars** (p. 74).

Parameters

window	Where it should be displayed
x	X Position of Star on scene
scale	How much bigger this star should be

Name: **Stars.cpp** (p. 105) Purpose: Class **Stars** (p. 74)

Author

Fenris

Version

0.88a 03/05/2017

5.17.2.2 ~Stars()

```
Stars::~Stars ( )  [inline]
```

Default destructor of **Stars** (p. 74).

5.17.3 Member Function Documentation

5.17.3.1 checkHitEdge()

```
void Stars::checkHitEdge ( )
```

Check if star gone to far.

5.17.3.2 draw()

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

Display stars on scene.

Parameters

window	Where to display them

5.17.3.3 drawStars()

```
void Stars::drawStars ( )
```

Draw stars.

5.17.3.4 getStars()

```
Constellation& Stars::getStars ( )  [inline]
```

Get the constellation.

5.17.3.5 getX()

```
unsigned Stars::getX ( ) const
```

Get X Position of star.

5.17.3.6 getY()

```
unsigned Stars::getY ( ) const
```

Get Y Position of star.

5.17.3.7 isHit()

```
bool Stars::isHit ( ) const  [inline]
```

See if star was hit.

5.17.3.8 move()

```
void Stars::move ( )
```

Move star.

5.17.3.9 removeHitStars()

```
void Stars::removeHitStars ( )
```

Remove stars that got hit.

5.17.3.10 setHit()

```
void Stars::setHit ( )  [inline]
```

Set that this star got hit.

5.17.3.11 updateStars()

```
void Stars::updateStars (
            sf::RenderWindow & window )
```

Update stars to get them moving and shooting.

Parameters

window	Where to display the stars

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Stars.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Stars.cpp**

5.18 Textures Class Reference

Class for **Textures** (p. 78).

```
#include <Textures.h>
```

Public Member Functions

- **Textures** ()

  *Default constructor for **Textures** (p. 78).*

Public Attributes

- sf::Texture **INVADER_11**

  Invader *(*p. *33) 1.*
- sf::Texture **INVADER_12**

  Texture for second frame of Invader1.
- sf::Texture **INVADER_13**

  Texture for third frame of Invader1.
- sf::Texture **INVADER_14**
- sf::Texture **INVADER_21**

  Invader *(*p. *33) 2.*
- sf::Texture **INVADER_22**

  Texture for second frame of Invader2.
- sf::Texture **INVADER_23**

  Texture for third frame of Invader2.
- sf::Texture **INVADER_24**
- sf::Texture **INVADER_31**

  Invader *(*p. *33) 3.*
- sf::Texture **INVADER_32**

  Texture for second frame of Invader3.
- sf::Texture **INVADER_33**

  Texture for third frame of Invader3.
- sf::Texture **INVADER_34**
- sf::Texture **INVADER_41**

  Invader *(*p. *33) 4.*
- sf::Texture **INVADER_42**

  Texture for second frame of Invader4.
- sf::Texture **INVADER_43**

  Texture for third frame of Invader4.
- sf::Texture **INVADER_44**
- sf::Texture **UFO_1**

  UFO.
- sf::Texture **UFO_2**

  Texture for second frame of UFO.
- sf::Texture **UFO_3**

  Texture for third frame of UFO.
- sf::Texture **UFO_4**
- sf::Texture **CREEPER_1**

  Creeper.
- sf::Texture **CREEPER_2**

  Texture for second frame of Creeper.
- sf::Texture **CREEPER_3**

  Texture for third frame of Creeper.
- sf::Texture **CREEPER_4**
- sf::Texture **TOUGH_1**

  Tough.
- sf::Texture **TOUGH_2**

  Texture for second frame of Tough.
- sf::Texture **TOUGH_3**

  Texture for third frame of Tough.
- sf::Texture **TOUGH_4**

- sf::Texture **BOSS_1**

  Boss.
- sf::Texture **BOSS_2**

  Texture for second frame of Boss.
- sf::Texture **BOSS_3**

  Texture for third frame of Boss.
- sf::Texture **BOSS_4**
- sf::Texture **SHIP_1**

  Ship.
- sf::Texture **SHIP_2**

  Texture for second frame of Ship.
- sf::Texture **SHIP_3**

  Texture for third frame of Ship.
- sf::Texture **SHIP_4**
- sf::Texture **EXPLOSION_1**

  Explosion *(*p. *26).*
- sf::Texture **EXPLOSION_2**

  Texture for second frame of ***Explosion*** *(*p. *26).*
- sf::Texture **EXPLOSION_3**
- sf::Texture **ARROW_1**

  Arrow.
- sf::Texture **ARROW_2**

  Texture for second frame of Arrow.
- sf::Texture **ARROW_3**

  Texture for third frame of Arrow.
- sf::Texture **ARROW_4**

  Texture for fourth frame of Arrow.
- sf::Texture **ARROW_5**

  Texture for fifth frame of Arrow.
- sf::Texture **ARROW_6**

  Texture for sixth frame of Arrow.
- sf::Texture **ARROW_7**

  Texture for seventh frame of Arrow.
- sf::Texture **ARROW_8**

  Texture for eight frame of Arrow.
- sf::Texture **ARROW_9**

  Texture for ninth frame of Arrow.
- sf::Texture **ARROW_10**

  Texture for tenth frame of Arrow.
- sf::Texture **ARROW_11**

  Texture for eleventh frame of Arrow.
- sf::Texture **ARROW_12**

  Texture for twelfth frame of Arrow.
- sf::Texture **ARROW_13**

  Texture for thirteenth frame of Arrow.
- sf::Texture **ARROW_14**

  Texture for fourteenth frame of Arrow.
- sf::Texture **ARROW_15**
- sf::Texture **BONUS_1**

  Bonuses.
- sf::Texture **BONUS_2**

Texture for Bonus2.

- sf::Texture **BONUS_3**

  Texture for Bonus3.
- sf::Texture **BONUS_4**

  Texture for Bonus4.
- sf::Texture **BONUS_5**

  Texture for Bonus5.
- sf::Texture **BONUS_6**

  Texture for Bonus6.
- sf::Texture **BONUS_7**

  Texture for Bonus7.
- sf::Texture **BONUS_8**

  Texture for Bonus8.

5.18.1 Detailed Description

Class for **Textures** (p. 78).

Name: **Textures.h** (p. 106) Purpose: Declaration of class **Textures** (p. 78)

Author

Fenris

Version

1.02a 14/05/2017

5.18.2 Constructor & Destructor Documentation

5.18.2.1 Textures()

```
Textures::Textures ( )
```

Default constructor for **Textures** (p. 78).

5.18.3 Member Data Documentation

5.18.3.1 ARROW_1

```
sf::Texture Textures::ARROW_1
```

Arrow.

Texture for first frame of Arrow

5.18.3.2 ARROW_10

`sf::Texture Textures::ARROW_10`

Texture for tenth frame of Arrow.

5.18.3.3 ARROW_11

`sf::Texture Textures::ARROW_11`

Texture for eleventh frame of Arrow.

5.18.3.4 ARROW_12

`sf::Texture Textures::ARROW_12`

Texture for twelfth frame of Arrow.

5.18.3.5 ARROW_13

`sf::Texture Textures::ARROW_13`

Texture for thirteenth frame of Arrow.

5.18.3.6 ARROW_14

`sf::Texture Textures::ARROW_14`

Texture for fourteenth frame of Arrow.

5.18.3.7 ARROW_15

`sf::Texture Textures::ARROW_15`

Texture for fifteenth frame of Arrow

5.18.3.8 ARROW_2

`sf::Texture Textures::ARROW_2`

Texture for second frame of Arrow.

5.18.3.9 ARROW_3

`sf::Texture Textures::ARROW_3`

Texture for third frame of Arrow.

5.18.3.10 ARROW_4

`sf::Texture Textures::ARROW_4`

Texture for fourth frame of Arrow.

5.18.3.11 ARROW_5

`sf::Texture Textures::ARROW_5`

Texture for fifth frame of Arrow.

5.18.3.12 ARROW_6

`sf::Texture Textures::ARROW_6`

Texture for sixth frame of Arrow.

5.18.3.13 ARROW_7

`sf::Texture Textures::ARROW_7`

Texture for seventh frame of Arrow.

5.18.3.14 ARROW_8

`sf::Texture Textures::ARROW_8`

Texture for eight frame of Arrow.

5.18.3.15 ARROW_9

`sf::Texture Textures::ARROW_9`

Texture for ninth frame of Arrow.

5.18.3.16 BONUS_1

`sf::Texture Textures::BONUS_1`

Bonuses.

Texture for Bonus1

5.18.3.17 BONUS_2

`sf::Texture Textures::BONUS_2`

Texture for Bonus2.

5.18.3.18 BONUS_3

`sf::Texture Textures::BONUS_3`

Texture for Bonus3.

5.18.3.19 BONUS_4

`sf::Texture Textures::BONUS_4`

Texture for Bonus4.

5.18.3.20 BONUS_5

`sf::Texture Textures::BONUS_5`

Texture for Bonus5.

5.18.3.21 BONUS_6

`sf::Texture Textures::BONUS_6`

Texture for Bonus6.

5.18.3.22 BONUS_7

`sf::Texture Textures::BONUS_7`

Texture for Bonus7.

5.18.3.23 BONUS_8

`sf::Texture Textures::BONUS_8`

Texture for Bonus8.

5.18.3.24 BOSS_1

`sf::Texture Textures::BOSS_1`

Boss.

Texture for first frame of Boss

5.18.3.25 BOSS_2

`sf::Texture Textures::BOSS_2`

Texture for second frame of Boss.

5.18.3.26 BOSS_3

```
sf::Texture Textures::BOSS_3
```

Texture for third frame of Boss.

5.18.3.27 BOSS_4

```
sf::Texture Textures::BOSS_4
```

Texture for fourth frame of Boss

5.18.3.28 CREEPER_1

```
sf::Texture Textures::CREEPER_1
```

Creeper.

Texture for first frame of Creeper

5.18.3.29 CREEPER_2

```
sf::Texture Textures::CREEPER_2
```

Texture for second frame of Creeper.

5.18.3.30 CREEPER_3

```
sf::Texture Textures::CREEPER_3
```

Texture for third frame of Creeper.

5.18.3.31 CREEPER_4

```
sf::Texture Textures::CREEPER_4
```

Texture for fourth frame of Creeper

5.18.3.32 EXPLOSION_1

```
sf::Texture Textures::EXPLOSION_1
```

Explosion (p. 26).

Texture for first frame of **Explosion** (p. 26)

5.18.3.33 EXPLOSION_2

```
sf::Texture Textures::EXPLOSION_2
```

Texture for second frame of **Explosion** (p. 26).

5.18.3.34 EXPLOSION_3

```
sf::Texture Textures::EXPLOSION_3
```

Texture for third frame of **Explosion** (p. 26)

5.18.3.35 INVADER_11

```
sf::Texture Textures::INVADER_11
```

Invader (p. 33) 1.

Texture for first frame of Invader1

5.18.3.36 INVADER_12

```
sf::Texture Textures::INVADER_12
```

Texture for second frame of Invader1.

5.18.3.37 INVADER_13

```
sf::Texture Textures::INVADER_13
```

Texture for third frame of Invader1.

5.18.3.38 INVADER_14

```
sf::Texture Textures::INVADER_14
```

Texture for fourth frame of Invader1

5.18.3.39 INVADER_21

```
sf::Texture Textures::INVADER_21
```

Invader (p. 33) 2.

Texture for first frame of Invader2

5.18.3.40 INVADER_22

```
sf::Texture Textures::INVADER_22
```

Texture for second frame of Invader2.

5.18.3.41 INVADER_23

```
sf::Texture Textures::INVADER_23
```

Texture for third frame of Invader2.

5.18.3.42 INVADER_24

```
sf::Texture Textures::INVADER_24
```

Texture for fourth frame of Invader2

5.18.3.43 INVADER_31

```
sf::Texture Textures::INVADER_31
```

Invader (p. 33) 3.

Texture for first frame of Invader3

5.18.3.44 INVADER_32

```
sf::Texture Textures::INVADER_32
```

Texture for second frame of Invader3.

5.18.3.45 INVADER_33

```
sf::Texture Textures::INVADER_33
```

Texture for third frame of Invader3.

5.18.3.46 INVADER_34

```
sf::Texture Textures::INVADER_34
```

Texture for fourth frame of Invader3

5.18.3.47 INVADER_41

```
sf::Texture Textures::INVADER_41
```

Invader (p. 33) 4.

Texture for first frame of Invader4

5.18.3.48 INVADER_42

```
sf::Texture Textures::INVADER_42
```

Texture for second frame of Invader4.

5.18.3.49 INVADER_43

```
sf::Texture Textures::INVADER_43
```

Texture for third frame of Invader4.

5.18.3.50 INVADER_44

```
sf::Texture Textures::INVADER_44
```

Texture for fourth frame of Invader4

5.18.3.51 SHIP_1

```
sf::Texture Textures::SHIP_1
```

Ship.

Texture for first frame of Ship

5.18.3.52 SHIP_2

```
sf::Texture Textures::SHIP_2
```

Texture for second frame of Ship.

5.18.3.53 SHIP_3

```
sf::Texture Textures::SHIP_3
```

Texture for third frame of Ship.

5.18.3.54 SHIP_4

```
sf::Texture Textures::SHIP_4
```

Texture for fourth frame of Ship

5.18.3.55 TOUGH_1

```
sf::Texture Textures::TOUGH_1
```

Tough.

Texture for first frame of Tough

5.18.3.56 TOUGH_2

`sf::Texture Textures::TOUGH_2`

Texture for second frame of Tough.

5.18.3.57 TOUGH_3

`sf::Texture Textures::TOUGH_3`

Texture for third frame of Tough.

5.18.3.58 TOUGH_4

`sf::Texture Textures::TOUGH_4`

Texture for fourth frame of Tough

5.18.3.59 UFO_1

`sf::Texture Textures::UFO_1`

UFO.

Texture for first frame of UFO

5.18.3.60 UFO_2

`sf::Texture Textures::UFO_2`

Texture for second frame of UFO.

5.18.3.61 UFO_3

`sf::Texture Textures::UFO_3`

Texture for third frame of UFO.

5.18.3.62 UFO_4

`sf::Texture Textures::UFO_4`

Texture for fourth frame of UFO

The documentation for this class was generated from the following files:

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Textures.h**
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ **Textures.cpp**

Chapter 6

File Documentation

6.1 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Bonus.cpp File Reference

```
#include "Bonus.h"
```

6.2 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Bonus.h File Reference

```
#include <SFML/Graphics.hpp>
#include "globals.h"
#include "Invader.h"
#include "Textures.h"
#include "LivesDisplay.h"
#include "PlayerLaser.h"
```

Classes

- class **Bonus**

  Class which consists of functions for displaying bonuses.

6.3 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ClockDisplay.cpp File Reference

```
#include <sstream>
#include "ClockDisplay.h"
#include "game.h"
```

6.4 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ClockDisplay.h File Reference

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

Classes

- class **ClockDisplay**
  Class for displaying clock.

6.5 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Explosions.cpp File Reference

```
#include "Explosions.h"
```

6.6 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Explosions.h File Reference

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

Classes

- class **Explosion**
  Class for one explosion.
- class **Explosions**
  Class for all explosions.

6.7 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/game.cpp File Reference

```
#include <fstream>
#include <sstream>
#include "game.h"
#include "globals.h"
```

6.8 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/game.h File Reference

```
#include <SFML/Graphics.hpp>
#include "ScoreDisplay.h"
#include "LivesDisplay.h"
#include "PlayerLaser.h"
#include "InvaderFormation.h"
#include "Spaceship.h"
#include "Menu.h"
#include "Explosions.h"
#include "Info.h"
#include "Load.h"
#include "Save.h"
#include "Menu_in_game.h"
#include "ClockDisplay.h"
#include "Stars.h"
```

Namespaces

- **Game**

  Namespace with logistic of the game.

Functions

- void **Game::handle_events** (sf::Window &window, sf::RenderWindow &windows, **Textures** &textures, **ClockDisplay** &clock_disp, **Info** &info, **Save** &save, **MenuIG** &menuig, **Load** &load, **Menu** &menu, **ScoreDisplay** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Spaceship** &ship, **Explosions** &explosions, unsigned &wave_on)

  Events(mainly window close)
- void **Game::real_time_key** (**PlayerLaser** &player_laser, **Spaceship** &ship)

  Real-time keyboard input.
- void **Game::update_objects** (sf::RenderWindow &window, **Stars** &stars, **MenuIG** &menuig, **Load** &load, **Save** &save, **Menu** &menu, **Textures** &textures, **Spaceship** &ship, **PlayerLaser** &player_laser, **Invader↩ Formation** &invaders, **LivesDisplay** &lives_disp, **Explosions** &explosions, unsigned &game_score, unsigned &wave_on, unsigned &minutes, unsigned &seconds, unsigned &miliseconds)

  Update all game objects.
- void **Game::draw_player_laser** (sf::RenderWindow &window, **PlayerLaser** &laser)

  Draw player lasers.
- void **Game::draw_text** (sf::RenderWindow &window, const std::string msg, const unsigned x, const unsigned y, sf::Color color=sf::Color::White, unsigned size=24)

  Draw text on screen.
- void **Game::draw_objects** (sf::RenderWindow &window, **Stars** &stars, **ClockDisplay** &clock_disp, **MenuIG** &menuig, **Load** &load, **Save** &save, **Info** &info, **Menu** &menu, **ScoreDisplay** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  Draw objects on screen.
- void **Game::handle_player_kill** (**InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Explosions** &explosions)

  An event after player was killed (mainly pause)
- void **Game::setup_wave** (sf::RenderWindow &window, **Textures** &textures, **InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Spaceship** &ship, **Explosions** &explosions, unsigned &wave_on, bool start↩ _game=false)

  Setup new wave.
- void **Game::draw_wave** (**InvaderFormation** &invaders)

  Draw wave on screen.
- void **Game::handle_game_over** (sf::RenderWindow &window, **ClockDisplay** &clock_disp, **Menu** &menu, **ScoreDisplay** &score_disp, **LivesDisplay** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  *An event after **Game** (p. 7) Over.*
- void **Game::reset_game** (**Menu** &menu, **ClockDisplay** &clock_disp, **ScoreDisplay** &score_disp, **Lives↩ Display** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  Reset all objects.
- void **Game::goto_menu** (**Menu** &menu, **ClockDisplay** &clock_disp, **ScoreDisplay** &score_disp, **Lives↩ Display** &lives_disp, **InvaderFormation** &invaders, **Spaceship** &ship, **PlayerLaser** &playerlaser, **Explosions** &explosions, unsigned &wave_on)

  Go to main menu.
- void **Game::goto_menuig** (**MenuIG** &menuig)

  Go to menu in game.
- void **Game::goto_info** (**Info** &info)

See info about Invaders.

- void **Game::goto_loading** (**Load** &load)

  Go to State for Loading game.

- void **Game::goto_saving** (**Save** &save)

  Go to State for Saving game.

- void **Game::play_game** (sf::RenderWindow &window, **Textures** &textures, **InvaderFormation** &invaders, **PlayerLaser** &player_laser, **Spaceship** &ship, **Explosions** &explosions, unsigned &wave_on)

  *Play New **Game** (p. 7).*

Variables

- bool **Game::life_awarded** = false

  Variable to see if life was awarded.

6.9 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/globals.cpp File Reference

```
#include "globals.h"
```

6.10 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/globals.h File Reference

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

Namespaces

- **Globals**

Enumerations

- enum **Globals::States** {
  Globals::MENU, **Globals::MENUIG**, **Globals::PLAY**, **Globals::WAVE_SETUP**,
  Globals::PLAYER_KILLED, **Globals::SAVE**, **Globals::LOAD**, **Globals::INFO**,
  Globals::GAME_OVER }

  States the game can be in.

Variables

- const std::string **Globals::SCREEN_TITLE** = "Space Invaders"

  Title of Window.
- constexpr unsigned **Globals::SCREEN_WIDTH** = 1366

  Width of Window.
- constexpr unsigned **Globals::SCREEN_HEIGHT** = 768

  Height of Window.
- constexpr unsigned **Globals::FRAME_RATE** = 60

  Frame rate.
- const sf::Color **Globals::BG_COLOR** = sf::Color(8,8,16)

  Color of the background.
- const std::string **Globals::SPRITES_PATH** = "sprites/"

  *Path to get sprites for **Textures** (p. 78).*
- const std::string **Globals::FONTS_PATH** = "fonts/"

  Path to get font.
- const std::string **Globals::SAVES_PATH** = "saves/"

  Path to get files for load and where will be saved new files.
- const std::string **Globals::LEVELS_PATH** = "levels/"

  Path to get files for loading levels.
- States **Globals::GAME_STATE** = **Globals::States::MENU**

  Variable to know in which state we are now.
- States **Globals::PREVIOUS_STATE** = **Globals::States::MENU**

  Variable to know in which state we were before.

6.11 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Info.cpp File Reference

```
#include <sstream>
#include "Info.h"
#include "game.h"
```

6.12 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Info.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Textures.h"
#include "ScoreDisplay.h"
#include "ClockDisplay.h"
```

Classes

- class **Info**

  Class of State INFO.

6.13 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Invader.cpp File Reference

```
#include "Invader.h"
```

6.14 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Invader.h File Reference

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

Classes

- class **Invader**

  *Class for **Invader** (p. 33).*

6.15 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderFormation.cpp File Reference

```
#include "globals.h"
#include "rand.h"
#include "InvaderFormation.h"
#include "Spaceship.h"
```

Typedefs

- using **Json** = nlohmann::json

6.15.1 Typedef Documentation

6.15.1.1 Json

```
using Json = nlohmann::json
```

Name: **InvaderFormation.cpp** (p. 98) Purpose: Class **InvaderFormation** (p. 39)

Author

Fenris

Version

0.98a 03/05/2017

6.16 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderFormation.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Invader.h"
#include "InvaderLaser.h"
#include "PlayerLaser.h"
#include "LivesDisplay.h"
#include "Explosions.h"
#include "Bonus.h"
#include <fstream>
#include "json.hpp"
```

Classes

- class **InvaderFormation**

  *Class of **InvaderFormation*** (p. 39).

Typedefs

- typedef std::vector< **Invader** * > **InvaderRow**

  Vector which has Invaders in it.
- typedef std::vector< **InvaderRow** > **InvaderVector2D**

  Vector which has InvadersRows in it.
- typedef std::vector< **InvaderLaser** * > **Lasers**

  Vector which has InvadersLasers in it.
- typedef std::vector< **Bonus** * > **Bonuses**

  Vector which has Bonuses in it.

6.16.1 Typedef Documentation

6.16.1.1 Bonuses

```
typedef std::vector< Bonus*>  Bonuses
```

Vector which has Bonuses in it.

6.16.1.2 InvaderRow

```
typedef std::vector< Invader*>  InvaderRow
```

Vector which has Invaders in it.

6.16.1.3 InvaderVector2D

```
typedef std::vector< InvaderRow>  InvaderVector2D
```

Vector which has InvadersRows in it.

6.16.1.4 Lasers

```
typedef std::vector< InvaderLaser*>  Lasers
```

Vector which has InvadersLasers in it.

6.17 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderLaser.cpp File Reference

```
#include "InvaderLaser.h"
```

6.18 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderLaser.h File Reference

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

Classes

- class **InvaderLaser**
  *Class for **InvaderLaser*** (p. 44).

6.19 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/LivesDisplay.cpp File Reference

```
#include <sstream>
#include "LivesDisplay.h"
#include "game.h"
```

6.20 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/LivesDisplay.h File Reference

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

Classes

- class **LivesDisplay**
  Class for Lives Display.

6.21 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Load.cpp File Reference

```
#include <sstream>
#include "Load.h"
#include "game.h"
```

6.22 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Load.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Textures.h"
#include "LivesDisplay.h"
#include "ScoreDisplay.h"
#include "ClockDisplay.h"
#include <fstream>
```

Classes

- class **Load**
  Class of State LOAD.

6.23 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/main.cpp File Reference

```
#include <ctime>
#include <SFML/Graphics.hpp>
#include "globals.h"
#include "rand.h"
#include "game.h"
#include "Textures.h"
#include "Stars.h"
```

Functions

- int **main** ()

6.23.1 Function Documentation

6.23.1.1 main()

```
int main ( )
```

Name: **main.cpp** (p. 101) Purpose: Main file

Author

Fenris

Version

0.94a 03/05/2017

6.24 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu.cpp File Reference

```
#include <sstream>
#include "Menu.h"
#include "game.h"
```

6.25 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Textures.h"
#include "ScoreDisplay.h"
#include "ClockDisplay.h"
```

Classes

- class **Menu**

  Class of State MENU.

6.26 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu_in_game.cpp File Reference

```
#include <sstream>
#include "Menu_in_game.h"
#include "game.h"
```

6.27 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu_in_game.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Textures.h"
#include "ScoreDisplay.h"
#include "ClockDisplay.h"
```

Classes

- class **MenuIG**

  Class of State MENUIG.

6.28 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/PlayerLaser.cpp File Reference

```
#include <cmath>
#include "PlayerLaser.h"
```

6.29 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/PlayerLaser.h File Reference

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

Classes

- class **PlayerLaser**

  *Class for **PlayerLaser** (p. 61).*

6.30 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/rand.cpp File Reference

```
#include <cstdlib>
#include "rand.h"
```

6.31 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/rand.h File Reference

Namespaces

- **Rand**

  *Namespace of **Rand** (p. 17).*

Functions

- unsigned **Rand::random** (const unsigned low, const unsigned high)

  Randomizing.

6.32 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Save.cpp File Reference

```
#include <sstream>
#include "Save.h"
#include "game.h"
```

6.33 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Save.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Textures.h"
#include "LivesDisplay.h"
#include "ScoreDisplay.h"
#include "ClockDisplay.h"
#include <fstream>
```

Classes

- class **Save**

  Class of State SAVE.

6.34 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ScoreDisplay.cpp File Reference

```
#include <sstream>
#include "ScoreDisplay.h"
#include "game.h"
```

6.35 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ScoreDisplay.h File Reference

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

Classes

- class **ScoreDisplay**

  Class for Score Display.

6.36 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Spaceship.cpp File Reference

```
#include "globals.h"
#include "game.h"
#include "Spaceship.h"
```

6.37 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Spaceship.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Textures.h"
#include "InvaderFormation.h"
#include "Explosions.h"
```

Classes

- class **Spaceship**

  *Class for **Spaceship*** (p. *71*)*.*

6.38 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Stars.cpp File Reference

```
#include "Stars.h"
```

6.39 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Stars.h File Reference

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

Classes

- class **Stars**

  *Class of **Stars*** (p. *74*)*.*

Typedefs

- typedef std::vector< **Stars** * > **Constellation**

  *Vector which has **Stars*** (p. *74*) *in it.*

6.39.1 Typedef Documentation

6.39.1.1 Constellation

```
typedef std::vector< Stars*>  Constellation
```

Vector which has **Stars** (p. 74) in it.

6.40 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Textures.cpp File Reference

```
#include <string>
#include "globals.h"
#include "Textures.h"
```

6.41 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Textures.h File Reference

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

Classes

- class **Textures**

  *Class for **Textures** (p. 78).*

Index