

Space Invaders

Generated by Doxygen 1.8.13

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Namespace Documentation	7
4.1	Game Namespace Reference	7
4.1.1	Detailed Description	8
4.1.2	Function Documentation	8
4.1.2.1	draw_objects()	9
4.1.2.2	draw_player_laser()	9
4.1.2.3	draw_text()	9
4.1.2.4	draw_wave()	9
4.1.2.5	goto_info()	10
4.1.2.6	goto_loading()	10
4.1.2.7	goto_menu()	10
4.1.2.8	goto_menuig()	10
4.1.2.9	goto_saving()	10
4.1.2.10	handle_events()	11
4.1.2.11	handle_game_over()	11
4.1.2.12	handle_player_kill()	12

4.1.2.13	play_game()	12
4.1.2.14	real_time_key()	12
4.1.2.15	reset_game()	12
4.1.2.16	setup_wave()	13
4.1.2.17	update_objects()	13
4.1.3	Variable Documentation	13
4.1.3.1	life_awarded	13
4.2	Globals Namespace Reference	14
4.2.1	Detailed Description	14
4.2.2	Enumeration Type Documentation	14
4.2.2.1	States	14
4.2.3	Variable Documentation	15
4.2.3.1	BG_COLOR	15
4.2.3.2	FONT_PATH	15
4.2.3.3	FRAME_RATE	15
4.2.3.4	GAME_STATE	16
4.2.3.5	LEVELS_PATH	16
4.2.3.6	PREVIOUS_STATE	16
4.2.3.7	SAVES_PATH	16
4.2.3.8	SCREEN_HEIGHT	16
4.2.3.9	SCREEN_TITLE	17
4.2.3.10	SCREEN_WIDTH	17
4.2.3.11	SPRITES_PATH	17
4.3	Rand Namespace Reference	17
4.3.1	Detailed Description	17
4.3.2	Function Documentation	17
4.3.2.1	random()	17

5	Class Documentation	19
5.1	Bonus Class Reference	19
5.1.1	Detailed Description	20
5.1.2	Member Enumeration Documentation	20
5.1.2.1	BonusType	20
5.1.3	Constructor & Destructor Documentation	20
5.1.3.1	Bonus()	21
5.1.3.2	~Bonus()	21
5.1.4	Member Function Documentation	21
5.1.4.1	checkCollide() [1/2]	21
5.1.4.2	checkCollide() [2/2]	22
5.1.4.3	draw()	22
5.1.4.4	getBonus()	22
5.1.4.5	getType()	23
5.1.4.6	getX()	23
5.1.4.7	getY()	23
5.1.4.8	isHit()	23
5.1.4.9	move()	23
5.1.4.10	setHit()	24
5.2	ClockDisplay Class Reference	24
5.2.1	Detailed Description	24
5.2.2	Constructor & Destructor Documentation	24
5.2.2.1	ClockDisplay()	24
5.2.3	Member Function Documentation	25
5.2.3.1	draw()	25
5.2.3.2	drawClock()	25
5.2.3.3	reset()	26
5.3	Explosion Class Reference	26
5.3.1	Detailed Description	26
5.3.2	Constructor & Destructor Documentation	26

5.3.2.1	Explosion()	27
5.3.3	Member Function Documentation	27
5.3.3.1	draw()	27
5.3.3.2	isShowing()	28
5.3.3.3	update()	28
5.4	Explosions Class Reference	28
5.4.1	Detailed Description	28
5.4.2	Constructor & Destructor Documentation	29
5.4.2.1	Explosions()	29
5.4.2.2	~Explosions()	29
5.4.3	Member Function Documentation	29
5.4.3.1	draw()	29
5.4.3.2	newExplosion()	29
5.4.3.3	reset()	30
5.4.3.4	update()	30
5.5	Info Class Reference	30
5.5.1	Detailed Description	31
5.5.2	Constructor & Destructor Documentation	31
5.5.2.1	Info()	31
5.5.3	Member Function Documentation	31
5.5.3.1	draw()	32
5.5.3.2	drawLine()	32
5.5.3.3	reset()	32
5.6	Invader Class Reference	33
5.6.1	Detailed Description	34
5.6.2	Member Enumeration Documentation	34
5.6.2.1	InvaderType	34
5.6.3	Constructor & Destructor Documentation	35
5.6.3.1	Invader()	35
5.6.4	Member Function Documentation	35

5.6.4.1	checkHitEdge()	35
5.6.4.2	decBonusesOnScreen()	35
5.6.4.3	decLasersOnScreen()	36
5.6.4.4	die()	36
5.6.4.5	dropDown()	36
5.6.4.6	getBonusesOnScreen()	36
5.6.4.7	getHeight()	36
5.6.4.8	getLasersOnScreen()	36
5.6.4.9	getLives()	37
5.6.4.10	getMoveDir()	37
5.6.4.11	getScoreValue()	37
5.6.4.12	getSprite()	37
5.6.4.13	incBonusesOnScreen()	37
5.6.4.14	incDeathTick()	37
5.6.4.15	incLasersOnScreen()	38
5.6.4.16	isDead()	38
5.6.4.17	isExploding()	38
5.6.4.18	isVisible()	38
5.6.4.19	move()	38
5.6.4.20	reset()	38
5.6.4.21	reverseDir()	39
5.6.4.22	setLives()	39
5.7	InvaderFormation Class Reference	39
5.7.1	Detailed Description	40
5.7.2	Constructor & Destructor Documentation	40
5.7.2.1	InvaderFormation()	40
5.7.2.2	~InvaderFormation()	41
5.7.3	Member Function Documentation	41
5.7.3.1	clearLevel()	41
5.7.3.2	draw()	41

5.7.3.3	drawBonuses()	41
5.7.3.4	drawLasers()	42
5.7.3.5	getBonuses()	42
5.7.3.6	getInvaders()	42
5.7.3.7	getLasers()	42
5.7.3.8	getNumKilled()	42
5.7.3.9	getTotal()	43
5.7.3.10	loadLevel()	43
5.7.3.11	removeBonuses()	43
5.7.3.12	removeHitBonuses()	43
5.7.3.13	removeLasers()	43
5.7.3.14	reset()	43
5.7.3.15	update()	44
5.8	InvaderLaser Class Reference	44
5.8.1	Detailed Description	45
5.8.2	Constructor & Destructor Documentation	45
5.8.2.1	InvaderLaser()	45
5.8.2.2	~InvaderLaser()	46
5.8.3	Member Function Documentation	46
5.8.3.1	checkCollide() [1/2]	46
5.8.3.2	checkCollide() [2/2]	46
5.8.3.3	draw()	47
5.8.3.4	getX()	47
5.8.3.5	getY()	47
5.8.3.6	isHit()	47
5.8.3.7	move()	47
5.8.3.8	setHit()	48
5.8.3.9	willHurt()	48
5.9	LivesDisplay Class Reference	48
5.9.1	Detailed Description	49

5.9.2	Constructor & Destructor Documentation	49
5.9.2.1	LivesDisplay()	49
5.9.2.2	~LivesDisplay()	49
5.9.3	Member Function Documentation	49
5.9.3.1	addLife()	49
5.9.3.2	draw()	50
5.9.3.3	getLives()	50
5.9.3.4	removeLife()	50
5.9.3.5	reset()	50
5.9.3.6	setLives()	50
5.10	Load Class Reference	51
5.10.1	Detailed Description	51
5.10.2	Constructor & Destructor Documentation	52
5.10.2.1	Load()	52
5.10.3	Member Function Documentation	52
5.10.3.1	doLoad()	52
5.10.3.2	draw()	53
5.10.3.3	drawLine()	53
5.10.3.4	reset()	53
5.10.3.5	update()	54
5.10.4	Member Data Documentation	54
5.10.4.1	fileName	54
5.11	Menu Class Reference	54
5.11.1	Detailed Description	55
5.11.2	Constructor & Destructor Documentation	55
5.11.2.1	Menu()	55
5.11.3	Member Function Documentation	55
5.11.3.1	draw()	56
5.11.3.2	drawLine()	56
5.11.3.3	getSelect()	56

5.11.3.4	reset()	57
5.11.3.5	setSelect()	57
5.11.3.6	update()	57
5.12	MenuIG Class Reference	57
5.12.1	Detailed Description	58
5.12.2	Constructor & Destructor Documentation	58
5.12.2.1	MenuIG()	58
5.12.3	Member Function Documentation	59
5.12.3.1	draw()	59
5.12.3.2	drawLine()	59
5.12.3.3	getSelect()	60
5.12.3.4	reset()	60
5.12.3.5	setSelect()	60
5.12.3.6	update()	60
5.13	PlayerLaser Class Reference	61
5.13.1	Detailed Description	62
5.13.2	Constructor & Destructor Documentation	62
5.13.2.1	PlayerLaser()	62
5.13.3	Member Function Documentation	62
5.13.3.1	getShape()	62
5.13.3.2	getShape2()	63
5.13.3.3	getSpeed()	63
5.13.3.4	getStop1()	63
5.13.3.5	getStop2()	63
5.13.3.6	isDouble()	63
5.13.3.7	isShooting()	63
5.13.3.8	isShooting2()	64
5.13.3.9	move()	64
5.13.3.10	nowDouble()	64
5.13.3.11	reset()	64

5.13.3.12	setSpeed()	64
5.13.3.13	shoot()	65
5.13.3.14	stop1()	65
5.13.3.15	stop2()	65
5.14	Save Class Reference	65
5.14.1	Detailed Description	66
5.14.2	Constructor & Destructor Documentation	66
5.14.2.1	Save()	66
5.14.3	Member Function Documentation	67
5.14.3.1	doSave()	67
5.14.3.2	draw()	67
5.14.3.3	drawLine()	68
5.14.3.4	reset()	68
5.14.3.5	update()	68
5.14.4	Member Data Documentation	69
5.14.4.1	fileName	69
5.15	ScoreDisplay Class Reference	69
5.15.1	Detailed Description	69
5.15.2	Constructor & Destructor Documentation	69
5.15.2.1	ScoreDisplay()	69
5.15.3	Member Function Documentation	70
5.15.3.1	draw()	70
5.15.3.2	drawScore()	70
5.15.3.3	reset()	71
5.16	Spaceship Class Reference	71
5.16.1	Detailed Description	71
5.16.2	Constructor & Destructor Documentation	72
5.16.2.1	Spaceship()	72
5.16.3	Member Function Documentation	72
5.16.3.1	die()	72

5.16.3.2	getSprite()	72
5.16.3.3	getWidth()	73
5.16.3.4	getX()	73
5.16.3.5	handleHit()	73
5.16.3.6	isHit()	73
5.16.3.7	move()	73
5.16.3.8	reset()	74
5.16.3.9	update()	74
5.17	Stars Class Reference	74
5.17.1	Detailed Description	75
5.17.2	Constructor & Destructor Documentation	75
5.17.2.1	Stars()	75
5.17.2.2	~Stars()	76
5.17.3	Member Function Documentation	76
5.17.3.1	checkHitEdge()	76
5.17.3.2	draw()	76
5.17.3.3	drawStars()	77
5.17.3.4	getStars()	77
5.17.3.5	getX()	77
5.17.3.6	getY()	77
5.17.3.7	isHit()	77
5.17.3.8	move()	77
5.17.3.9	removeHitStars()	78
5.17.3.10	setHit()	78
5.17.3.11	updateStars()	78
5.18	Textures Class Reference	78
5.18.1	Detailed Description	81
5.18.2	Constructor & Destructor Documentation	81
5.18.2.1	Textures()	81
5.18.3	Member Data Documentation	81

5.18.3.1	ARROW_1	81
5.18.3.2	ARROW_10	82
5.18.3.3	ARROW_11	82
5.18.3.4	ARROW_12	82
5.18.3.5	ARROW_13	82
5.18.3.6	ARROW_14	82
5.18.3.7	ARROW_15	82
5.18.3.8	ARROW_2	83
5.18.3.9	ARROW_3	83
5.18.3.10	ARROW_4	83
5.18.3.11	ARROW_5	83
5.18.3.12	ARROW_6	83
5.18.3.13	ARROW_7	83
5.18.3.14	ARROW_8	84
5.18.3.15	ARROW_9	84
5.18.3.16	BONUS_1	84
5.18.3.17	BONUS_2	84
5.18.3.18	BONUS_3	84
5.18.3.19	BONUS_4	84
5.18.3.20	BONUS_5	85
5.18.3.21	BONUS_6	85
5.18.3.22	BONUS_7	85
5.18.3.23	BONUS_8	85
5.18.3.24	BOSS_1	85
5.18.3.25	BOSS_2	85
5.18.3.26	BOSS_3	86
5.18.3.27	BOSS_4	86
5.18.3.28	CREEPER_1	86
5.18.3.29	CREEPER_2	86
5.18.3.30	CREEPER_3	86

5.18.3.31 CREEPER_4	86
5.18.3.32 EXPLOSION_1	87
5.18.3.33 EXPLOSION_2	87
5.18.3.34 EXPLOSION_3	87
5.18.3.35 INVADER_11	87
5.18.3.36 INVADER_12	87
5.18.3.37 INVADER_13	87
5.18.3.38 INVADER_14	88
5.18.3.39 INVADER_21	88
5.18.3.40 INVADER_22	88
5.18.3.41 INVADER_23	88
5.18.3.42 INVADER_24	88
5.18.3.43 INVADER_31	88
5.18.3.44 INVADER_32	89
5.18.3.45 INVADER_33	89
5.18.3.46 INVADER_34	89
5.18.3.47 INVADER_41	89
5.18.3.48 INVADER_42	89
5.18.3.49 INVADER_43	89
5.18.3.50 INVADER_44	90
5.18.3.51 SHIP_1	90
5.18.3.52 SHIP_2	90
5.18.3.53 SHIP_3	90
5.18.3.54 SHIP_4	90
5.18.3.55 TOUGH_1	90
5.18.3.56 TOUGH_2	91
5.18.3.57 TOUGH_3	91
5.18.3.58 TOUGH_4	91
5.18.3.59 UFO_1	91
5.18.3.60 UFO_2	91
5.18.3.61 UFO_3	91
5.18.3.62 UFO_4	91

6 File Documentation	93
6.1 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Bonus.cpp File Reference	93
6.2 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Bonus.h File Reference	93
6.3 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ClockDisplay.cpp File Reference	93
6.4 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ClockDisplay.h File Reference	93
6.5 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Explosions.cpp File Reference	94
6.6 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Explosions.h File Reference	94
6.7 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/game.cpp File Reference	94
6.8 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/game.h File Reference	94
6.9 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/globals.cpp File Reference	96
6.10 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/globals.h File Reference	96
6.11 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Info.cpp File Reference	97
6.12 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Info.h File Reference	97
6.13 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Invader.cpp File Reference	98
6.14 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Invader.h File Reference	98
6.15 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderFormation.cpp File Reference	98
6.15.1 Typedef Documentation	98
6.15.1.1 Json	98
6.16 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderFormation.h File Reference	99
6.16.1 Typedef Documentation	99
6.16.1.1 Bonuses	99
6.16.1.2 InvaderRow	99
6.16.1.3 InvaderVector2D	100
6.16.1.4 Lasers	100
6.17 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderLaser.cpp File Reference	100
6.18 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/InvaderLaser.h File Reference	100
6.19 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/LivesDisplay.cpp File Reference	100
6.20 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/LivesDisplay.h File Reference	100
6.21 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Load.cpp File Reference	101
6.22 E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Load.h File Reference	101

6.23	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/main.cpp File Reference	101
6.23.1	Function Documentation	101
6.23.1.1	main()	102
6.24	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu.cpp File Reference	102
6.25	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu.h File Reference	102
6.26	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu_in_game.cpp File Reference	102
6.27	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Menu_in_game.h File Reference	103
6.28	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/PlayerLaser.cpp File Reference	103
6.29	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/PlayerLaser.h File Reference	103
6.30	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/rand.cpp File Reference	103
6.31	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/rand.h File Reference	103
6.32	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Save.cpp File Reference	104
6.33	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Save.h File Reference	104
6.34	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ScoreDisplay.cpp File Reference	104
6.35	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ScoreDisplay.h File Reference	104
6.36	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Spaceship.cpp File Reference	105
6.37	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Spaceship.h File Reference	105
6.38	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Stars.cpp File Reference	105
6.39	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Stars.h File Reference	105
6.39.1	Typedef Documentation	106
6.39.1.1	Constellation	106
6.40	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Textures.cpp File Reference	106
6.41	E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/Textures.h File Reference	106

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Game		
	Namespace with logistic of the game	7
Globals	14
Rand		
	Namespace of Rand (p. 17)	17

Chapter 2

Class Index

2.1 Class List

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

Bonus	
Class which consists of functions for displaying bonuses	19
ClockDisplay	
Class for displaying clock	24
Explosion	
Class for one explosion	26
Explosions	
Class for all explosions	28
Info	
Class of State INFO	30
Invader	
Class for Invader (p. 33)	33
InvaderFormation	
Class of InvaderFormation (p. 39)	39
InvaderLaser	
Class for InvaderLaser (p. 44)	44
LivesDisplay	
Class for Lives Display	48
Load	
Class of State LOAD	51
Menu	
Class of State MENU	54
MenuIG	
Class of State MENUIG	57
PlayerLaser	
Class for PlayerLaser (p. 61)	61
Save	
Class of State SAVE	65
ScoreDisplay	
Class for Score Display	69
Spaceship	
Class for Spaceship (p. 71)	71
Stars	
Class of Stars (p. 74)	74
Textures	
Class for Textures (p. 78)	78

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Bonus.cpp	93
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Bonus.h	93
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ ClockDisplay.cpp	93
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ ClockDisplay.h	93
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Explosions.cpp	94
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Explosions.h	94
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ game.cpp	94
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ game.h	94
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ globals.cpp	96
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ globals.h	96
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Info.cpp	97
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Info.h	97
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Invader.cpp	98
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Invader.h	98
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ InvaderFormation.cpp	98
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ InvaderFormation.h	99
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ InvaderLaser.cpp	100
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ InvaderLaser.h	100
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ LivesDisplay.cpp	100
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ LivesDisplay.h	100
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Load.cpp	101
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Load.h	101
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ main.cpp	101
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Menu.cpp	102
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Menu.h	102
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Menu_in_game.cpp	102
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Menu_in_game.h	103
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ PlayerLaser.cpp	103
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ PlayerLaser.h	103
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ rand.cpp	103
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ rand.h	103
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Save.cpp	104
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Save.h	104
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ ScoreDisplay.cpp	104
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ ScoreDisplay.h	104

E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Spaceship.cpp	105
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Spaceship.h	105
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Stars.cpp	105
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Stars.h	105
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Textures.cpp	106
E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/ Textures.h	106

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, **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 **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, **InvaderFormation** &invaders, **LivesDisplay** &lives_disp, **Explosions** &explosions, unsigned &game_score, unsigned &wave_on, unsigned &minutes, unsigned &seconds, unsigned &milliseconds)
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** &player_laser, **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, **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 **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

Fennis

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 & milliseconds )
```

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

Fennis

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

<i>low</i>	The bottom line for randomizing
<i>high</i>	The upper line for randomizing

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

Author

Fennis

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

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

<i>x</i>	X Position of players ship
<i>y</i>	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

<i>rect</i>	Rectangle of players ship
-------------	---------------------------

5.1.4.3 `draw()`

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

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

Parameters

<i>window</i>	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

<i>type</i>	Type of Bonus (p. 19)
<i>lives</i>	For operations on lives
<i>game_score</i>	For operations on game score
<i>laser</i>	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 &milliseconds)
*Default constructor for **ClockDisplay** (p. 24).*
- void **drawClock** (sf::RenderWindow &window, const unsigned minutes, const unsigned seconds, const unsigned milliseconds, 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 milliseconds 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 & milliseconds )
```

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

Parameters

<i>minutes</i>	Variable for minutes to display on Clock
<i>seconds</i>	Variable for seconds to display on Clock
<i>milliseconds</i>	Variable for milliseconds 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 milliseconds,
    const unsigned x,
    const unsigned y )
```

Get text to display on scene.

Parameters

<i>window</i>	Where it should be drawn
<i>minutes</i>	Variable for minutes to display on Clock
<i>seconds</i>	Variable for seconds to display on Clock
<i>milliseconds</i>	Variable for milliseconds to display on Clock
<i>x</i>	X Position of ClockDisplay (p. 24) on scene
<i>y</i>	Y Position of ClockDisplay (p. 24) on scene

5.2.3.3 reset()

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

Reset minutes, seconds and milliseconds 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

<i>textures</i>	Texture of Explosion (p. 26)
<i>color</i>	Color of Explosion (p. 26)
<i>x</i>	X Position of Explosion (p. 26) on scene
<i>y</i>	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

<i>window</i>	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

<i>textures</i>	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

<i>window</i>	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

<i>color</i>	Color of new Explosion (p. 26)
<i>x</i>	X Position for new Explosion (p. 26)
<i>y</i>	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

<i>textures</i>	Textures (p. 78) which will be displayed next to score
<i>score_disp</i>	Score displayed on scene
<i>clock_disp</i>	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

<i>window</i>	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

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

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

If it's okay it will draw 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 much 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

<i>textures</i>	Texture for Invader (p. 33)'s sprite
<i>type</i>	Type of Invader (p. 33)
<i>isVisible</i>	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

<i>screenw</i>	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 much 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

<i>x</i>	X Position for Invader (p. 33)
<i>y</i>	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
&milliseconds)
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

<i>window</i>	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

<i>amount</i>	
---------------	--

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 & milliseconds )
```

Update the formation and it's movement.

Parameters

<i>laser</i>	For updating movement
<i>ship</i>	For updating movement
<i>player_laser</i>	For updating lasers
<i>lives_disp</i>	For checking if there was a hit
<i>explosions</i>	For updating lasers
<i>game_score</i>	For updating score
<i>minutes</i>	For updating clock
<i>seconds</i>	For updating clock
<i>milliseconds</i>	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

<i>x</i>	X Position of the laser
<i>y</i>	Y Position of the laser
<i>will_hurt</i>	Whether or not the laser will hurt the player
<i>owner</i>	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

<i>x</i>	X Position of players ship
<i>y</i>	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

<i>rect</i>	Rectangle of players ship
-------------	---------------------------

5.8.3.3 draw()

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

Draw laser on the scene.

Parameters

<i>window</i>	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

<i>textures</i>	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

<i>window</i>	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

<i>num</i>	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 &milliseconds)
*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 & milliseconds )
```

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

Parameters

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

<i>window</i>	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

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

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

If it's okay it will draw 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

<i>textures</i>	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

<i>textures</i>	Textures (p. 78) which will be displayed next to score
<i>score_disp</i>	Score displayed on scene
<i>clock_disp</i>	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

<i>window</i>	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

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

<i>select</i>	Option which will be selected
---------------	-------------------------------

5.11.3.6 update()

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

Update arrow to get new frame.

Parameters

<i>textures</i>	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

<i>textures</i>	Textures (p. 78) which will be displayed next to score
<i>score_disp</i>	Score displayed on scene
<i>clock_disp</i>	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

<i>window</i>	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

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

Parameters

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

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

If it's okay it will draw 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

<i>select</i>	Option which will be selected
---------------	-------------------------------

5.12.3.6 `update()`

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

Update arrow to get new frame.

Parameters

<i>textures</i>	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

<i>explosions</i>	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

<i>speed</i>	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

<i>startx</i>	X Position of the laser that will be displayed
<i>starty</i>	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 &milliseconds)
*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 & milliseconds )
```

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

Parameters

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

<i>window</i>	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

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

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

If it's okay it will draw 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

<i>textures</i>	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

<i>score</i>	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

<i>window</i>	Where it should be drawn
<i>score</i>	Score to be displayed
<i>x</i>	X Position of ScoreDisplay (p. 69) on scene
<i>y</i>	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

<i>textures</i>	Textures (p. 78) that will be displayed on scene
<i>startx</i>	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

<i>invaders</i>	For removal of theirs lasers
<i>player_laser</i>	To stop all players lasers
<i>explosions</i>	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

<i>invaders</i>	For removal of theirs lasers
<i>player_laser</i>	To stop all player lasers and to speed up or decelerate them
<i>explosions</i>	To remove all explosions
<i>lives</i>	To change lives
<i>game_score</i>	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

<i>dir</i>	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

<i>invaders</i>	For removal of theirs lasers
<i>player_laser</i>	To stop all player lasers and to speed up or decelerate them
<i>explosions</i>	To remove all explosions
<i>lives_disp</i>	To change lives
<i>game_score</i>	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

<i>window</i>	Where it should be displayed
<i>x</i>	X Position of Star on scene
<i>scale</i>	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

<i>window</i>	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

<i>window</i>	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, **InvaderFormation** &invaders, **LivesDisplay** &lives_disp, **Explosions** &explosions, unsigned &game_score, unsigned &wave_on, unsigned &minutes, unsigned &seconds, unsigned &milliseconds)
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, **LivesDisplay** &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, **LivesDisplay** &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

- ~Bonus
 - Bonus, 21
- ~Explosions
 - Explosions, 29
- ~InvaderFormation
 - InvaderFormation, 41
- ~InvaderLaser
 - InvaderLaser, 46
- ~LivesDisplay
 - LivesDisplay, 49
- ~Stars
 - Stars, 76

- ARROW_1
 - Textures, 81
- ARROW_10
 - Textures, 81
- ARROW_11
 - Textures, 82
- ARROW_12
 - Textures, 82
- ARROW_13
 - Textures, 82
- ARROW_14
 - Textures, 82
- ARROW_15
 - Textures, 82
- ARROW_2
 - Textures, 82
- ARROW_3
 - Textures, 83
- ARROW_4
 - Textures, 83
- ARROW_5
 - Textures, 83
- ARROW_6
 - Textures, 83
- ARROW_7
 - Textures, 83
- ARROW_8
 - Textures, 83
- ARROW_9
 - Textures, 84
- addLife
 - LivesDisplay, 49

- BG_COLOR
 - Globals, 15
- BONUS_1
 - Textures, 84

- BONUS_2
 - Textures, 84
- BONUS_3
 - Textures, 84
- BONUS_4
 - Textures, 84
- BONUS_5
 - Textures, 84
- BONUS_6
 - Textures, 85
- BONUS_7
 - Textures, 85
- BONUS_8
 - Textures, 85
- BOSS_1
 - Textures, 85
- BOSS_2
 - Textures, 85
- BOSS_3
 - Textures, 85
- BOSS_4
 - Textures, 86
- Bonus, 19
 - ~Bonus, 21
 - Bonus, 20
 - BonusType, 20
 - checkCollide, 21, 22
 - draw, 22
 - getBonus, 22
 - getType, 23
 - getX, 23
 - getY, 23
 - isHit, 23
 - move, 23
 - setHit, 23
- BonusType
 - Bonus, 20
- Bonuses
 - InvaderFormation.h, 99
- CREEPER_1
 - Textures, 86
- CREEPER_2
 - Textures, 86
- CREEPER_3
 - Textures, 86
- CREEPER_4
 - Textures, 86
- checkCollide
 - Bonus, 21, 22

- InvaderLaser, 46
- checkHitEdge
 - Invader, 35
 - Stars, 76
- clearLevel
 - InvaderFormation, 41
- ClockDisplay, 24
 - ClockDisplay, 24
 - draw, 25
 - drawClock, 25
 - reset, 26
- Constellation
 - Stars.h, 106
- decBonusesOnScreen
 - Invader, 35
- decLasersOnScreen
 - Invader, 35
- die
 - Invader, 36
 - Spaceship, 72
- doLoad
 - Load, 52
- doSave
 - Save, 67
- draw
 - Bonus, 22
 - ClockDisplay, 25
 - Explosion, 27
 - Explosions, 29
 - Info, 31
 - InvaderFormation, 41
 - InvaderLaser, 47
 - LivesDisplay, 49
 - Load, 52
 - Menu, 55
 - MenuIG, 59
 - Save, 67
 - ScoreDisplay, 70
 - Stars, 76
- draw_objects
 - Game, 8
- draw_player_laser
 - Game, 9
- draw_text
 - Game, 9
- draw_wave
 - Game, 9
- drawBonuses
 - InvaderFormation, 41
- drawClock
 - ClockDisplay, 25
- drawLasers
 - InvaderFormation, 41
- drawLine
 - Info, 32
 - Load, 53
 - Menu, 56
 - MenuIG, 59
 - Save, 67
- drawScore
 - ScoreDisplay, 70
- drawStars
 - Stars, 77
- dropDown
 - Invader, 36
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Bonus.cpp, 93
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Bonus.h, 93
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - ClockDisplay.cpp, 93
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - ClockDisplay.h, 93
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Explosions.cpp, 94
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Explosions.h, 94
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Info.cpp, 97
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Info.h, 97
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Invader.cpp, 98
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Invader.h, 98
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - InvaderFormation.cpp, 98
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - InvaderFormation.h, 99
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - InvaderLaser.cpp, 100
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - InvaderLaser.h, 100
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - LivesDisplay.cpp, 100
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - LivesDisplay.h, 100
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Load.cpp, 101
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Load.h, 101
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Menu.cpp, 102
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Menu.h, 102
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Menu_in_game.cpp, 102
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Menu_in_game.h, 103
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - PlayerLaser.cpp, 103
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - PlayerLaser.h, 103
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↵
 - Save.cpp, 104

- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Save.h, 104
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ ScoreDisplay.cpp, 104
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ ScoreDisplay.h, 104
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Spaceship.cpp, 105
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Spaceship.h, 105
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Stars.cpp, 105
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Stars.h, 105
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Textures.cpp, 106
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/↔ Textures.h, 106
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/game.↔ cpp, 94
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/game.↔ h, 94
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/globals.↔ cpp, 96
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/globals.↔ h, 96
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/main.↔ cpp, 101
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/rand.↔ cpp, 103
- E:/Piotrek/Informatyka/C++/Studia/PROI-v31 — kopia/rand.↔ h, 103
- EXPLOSION_1
 - Textures, 86
- EXPLOSION_2
 - Textures, 87
- EXPLOSION_3
 - Textures, 87
- Explosion, 26
 - draw, 27
 - Explosion, 26
 - isShowing, 27
 - update, 28
- Explosions, 28
 - ~Explosions, 29
 - draw, 29
 - Explosions, 29
 - newExplosion, 29
 - reset, 30
 - update, 30
- FONTS_PATH
 - Globals, 15
- FRAME_RATE
 - Globals, 15
- fileName
 - Load, 54
 - Save, 69
- GAME_STATE
 - Globals, 15
- Game, 7
 - draw_objects, 8
 - draw_player_laser, 9
 - draw_text, 9
 - draw_wave, 9
 - goto_info, 9
 - goto_loading, 10
 - goto_menu, 10
 - goto_menuig, 10
 - goto_saving, 10
 - handle_events, 10
 - handle_game_over, 11
 - handle_player_kill, 11
 - life_awarded, 13
 - play_game, 12
 - real_time_key, 12
 - reset_game, 12
 - setup_wave, 12
 - update_objects, 13
- getBonus
 - Bonus, 22
- getBonuses
 - InvaderFormation, 42
- getBonusesOnScreen
 - Invader, 36
- getHeight
 - Invader, 36
- getInvaders
 - InvaderFormation, 42
- getLasers
 - InvaderFormation, 42
- getLasersOnScreen
 - Invader, 36
- getLives
 - Invader, 36
 - LivesDisplay, 50
- getMoveDir
 - Invader, 37
- getNumKilled
 - InvaderFormation, 42
- getScoreValue
 - Invader, 37
- getSelect
 - Menu, 56
 - MenuIG, 60
- getShape
 - PlayerLaser, 62
- getShape2
 - PlayerLaser, 62
- getSpeed
 - PlayerLaser, 63
- getSprite
 - Invader, 37
 - Spaceship, 72
- getStars
 - Stars, 77

- getStop1
 - PlayerLaser, 63
- getStop2
 - PlayerLaser, 63
- getTotal
 - InvaderFormation, 42
- getType
 - Bonus, 23
- getWidth
 - Spaceship, 72
- getX
 - Bonus, 23
 - InvaderLaser, 47
 - Spaceship, 73
 - Stars, 77
- getY
 - Bonus, 23
 - InvaderLaser, 47
 - Stars, 77
- Globals, 14
 - BG_COLOR, 15
 - FONTS_PATH, 15
 - FRAME_RATE, 15
 - GAME_STATE, 15
 - LEVELS_PATH, 16
 - PREVIOUS_STATE, 16
 - SAVES_PATH, 16
 - SCREEN_HEIGHT, 16
 - SCREEN_TITLE, 16
 - SCREEN_WIDTH, 17
 - SPRITES_PATH, 17
 - States, 14
- goto_info
 - Game, 9
- goto_loading
 - Game, 10
- goto_menu
 - Game, 10
- goto_menuig
 - Game, 10
- goto_saving
 - Game, 10
- handle_events
 - Game, 10
- handle_game_over
 - Game, 11
- handle_player_kill
 - Game, 11
- handleHit
 - Spaceship, 73
- INVADER_11
 - Textures, 87
- INVADER_12
 - Textures, 87
- INVADER_13
 - Textures, 87
- INVADER_14
 - Textures, 87
- INVADER_21
 - Textures, 88
- INVADER_22
 - Textures, 88
- INVADER_23
 - Textures, 88
- INVADER_24
 - Textures, 88
- INVADER_31
 - Textures, 88
- INVADER_32
 - Textures, 88
- INVADER_33
 - Textures, 89
- INVADER_34
 - Textures, 89
- INVADER_41
 - Textures, 89
- INVADER_42
 - Textures, 89
- INVADER_43
 - Textures, 89
- INVADER_44
 - Textures, 89
- incBonusesOnScreen
 - Invader, 37
- incDeathTick
 - Invader, 37
- incLasersOnScreen
 - Invader, 37
- Info, 30
 - draw, 31
 - drawLine, 32
 - Info, 31
 - reset, 32
- Invader, 33
 - checkHitEdge, 35
 - decBonusesOnScreen, 35
 - decLasersOnScreen, 35
 - die, 36
 - dropDown, 36
 - getBonusesOnScreen, 36
 - getHeight, 36
 - getLasersOnScreen, 36
 - getLives, 36
 - getMoveDir, 37
 - getScoreValue, 37
 - getSprite, 37
 - incBonusesOnScreen, 37
 - incDeathTick, 37
 - incLasersOnScreen, 37
 - Invader, 35
 - InvaderType, 34
 - isDead, 38
 - isExploding, 38
 - isVisible, 38
 - move, 38

- reset, 38
- reverseDir, 39
- setLives, 39
- InvaderFormation, 39
 - ~InvaderFormation, 41
 - clearLevel, 41
 - draw, 41
 - drawBonuses, 41
 - drawLasers, 41
 - getBonuses, 42
 - getInvaders, 42
 - getLasers, 42
 - getNumKilled, 42
 - getTotal, 42
 - InvaderFormation, 40
 - loadLevel, 43
 - removeBonuses, 43
 - removeHitBonuses, 43
 - removeLasers, 43
 - reset, 43
 - update, 43
- InvaderFormation.cpp
 - Json, 98
- InvaderFormation.h
 - Bonuses, 99
 - InvaderRow, 99
 - InvaderVector2D, 99
 - Lasers, 100
- InvaderLaser, 44
 - ~InvaderLaser, 46
 - checkCollide, 46
 - draw, 47
 - getX, 47
 - getY, 47
 - InvaderLaser, 45
 - isHit, 47
 - move, 47
 - setHit, 47
 - willHurt, 48
- InvaderRow
 - InvaderFormation.h, 99
- InvaderType
 - Invader, 34
- InvaderVector2D
 - InvaderFormation.h, 99
- isDead
 - Invader, 38
- isDouble
 - PlayerLaser, 63
- isExploding
 - Invader, 38
- isHit
 - Bonus, 23
 - InvaderLaser, 47
 - Spaceship, 73
 - Stars, 77
- isShooting
 - PlayerLaser, 63
- isShooting2
 - PlayerLaser, 63
- isShowing
 - Explosion, 27
- isVisible
 - Invader, 38
- Json
 - InvaderFormation.cpp, 98
- LEVELS_PATH
 - Globals, 16
- Lasers
 - InvaderFormation.h, 100
- life_awarded
 - Game, 13
- LivesDisplay, 48
 - ~LivesDisplay, 49
 - addLife, 49
 - draw, 49
 - getLives, 50
 - LivesDisplay, 49
 - removeLife, 50
 - reset, 50
 - setLives, 50
- Load, 51
 - doLoad, 52
 - draw, 52
 - drawLine, 53
 - fileName, 54
 - Load, 52
 - reset, 53
 - update, 53
- loadLevel
 - InvaderFormation, 43
- main
 - main.cpp, 101
- main.cpp
 - main, 101
- Menu, 54
 - draw, 55
 - drawLine, 56
 - getSelect, 56
 - Menu, 55
 - reset, 56
 - setSelect, 57
 - update, 57
- MenuIG, 57
 - draw, 59
 - drawLine, 59
 - getSelect, 60
 - MenuIG, 58
 - reset, 60
 - setSelect, 60
 - update, 60
- move
 - Bonus, 23
 - Invader, 38

- InvaderLaser, 47
 - PlayerLaser, 64
 - Spaceship, 73
 - Stars, 77
- newExplosion
 - Explosions, 29
- nowDouble
 - PlayerLaser, 64
- PREVIOUS_STATE
 - Globals, 16
- play_game
 - Game, 12
- PlayerLaser, 61
 - getShape, 62
 - getShape2, 62
 - getSpeed, 63
 - getStop1, 63
 - getStop2, 63
 - isDouble, 63
 - isShooting, 63
 - isShooting2, 63
 - move, 64
 - nowDouble, 64
 - PlayerLaser, 62
 - reset, 64
 - setSpeed, 64
 - shoot, 65
 - stop1, 65
 - stop2, 65
- Rand, 17
 - random, 17
- random
 - Rand, 17
- real_time_key
 - Game, 12
- removeBonuses
 - InvaderFormation, 43
- removeHitBonuses
 - InvaderFormation, 43
- removeHitStars
 - Stars, 77
- removeLasers
 - InvaderFormation, 43
- removeLife
 - LivesDisplay, 50
- reset
 - ClockDisplay, 26
 - Explosions, 30
 - Info, 32
 - Invader, 38
 - InvaderFormation, 43
 - LivesDisplay, 50
 - Load, 53
 - Menu, 56
 - MenuIG, 60
 - PlayerLaser, 64
 - Save, 68
 - ScoreDisplay, 70
 - Spaceship, 74
- reset_game
 - Game, 12
- reverseDir
 - Invader, 39
- SAVES_PATH
 - Globals, 16
- SCREEN_HEIGHT
 - Globals, 16
- SCREEN_TITLE
 - Globals, 16
- SCREEN_WIDTH
 - Globals, 17
- SHIP_1
 - Textures, 90
- SHIP_2
 - Textures, 90
- SHIP_3
 - Textures, 90
- SHIP_4
 - Textures, 90
- SPRITES_PATH
 - Globals, 17
- Save, 65
 - doSave, 67
 - draw, 67
 - drawLine, 67
 - fileName, 69
 - reset, 68
 - Save, 66
 - update, 68
- ScoreDisplay, 69
 - draw, 70
 - drawScore, 70
 - reset, 70
 - ScoreDisplay, 69
- setHit
 - Bonus, 23
 - InvaderLaser, 47
 - Stars, 78
- setLives
 - Invader, 39
 - LivesDisplay, 50
- setSelect
 - Menu, 57
 - MenuIG, 60
- setSpeed
 - PlayerLaser, 64
- setup_wave
 - Game, 12
- shoot
 - PlayerLaser, 65
- Spaceship, 71
 - die, 72
 - getSprite, 72
 - getWidth, 72

- getX, 73
- handleHit, 73
- isHit, 73
- move, 73
- reset, 74
- Spaceship, 72
- update, 74
- Stars, 74
 - ~Stars, 76
 - checkHitEdge, 76
 - draw, 76
 - drawStars, 77
 - getStars, 77
 - getX, 77
 - getY, 77
 - isHit, 77
 - move, 77
 - removeHitStars, 77
 - setHit, 78
 - Stars, 75
 - updateStars, 78
- Stars.h
 - Constellation, 106
- States
 - Globals, 14
- stop1
 - PlayerLaser, 65
- stop2
 - PlayerLaser, 65
- TOUGH_1
 - Textures, 90
- TOUGH_2
 - Textures, 90
- TOUGH_3
 - Textures, 91
- TOUGH_4
 - Textures, 91
- Textures, 78
 - ARROW_1, 81
 - ARROW_10, 81
 - ARROW_11, 82
 - ARROW_12, 82
 - ARROW_13, 82
 - ARROW_14, 82
 - ARROW_15, 82
 - ARROW_2, 82
 - ARROW_3, 83
 - ARROW_4, 83
 - ARROW_5, 83
 - ARROW_6, 83
 - ARROW_7, 83
 - ARROW_8, 83
 - ARROW_9, 84
 - BONUS_1, 84
 - BONUS_2, 84
 - BONUS_3, 84
 - BONUS_4, 84
 - BONUS_5, 84
 - BONUS_6, 85
 - BONUS_7, 85
 - BONUS_8, 85
 - BOSS_1, 85
 - BOSS_2, 85
 - BOSS_3, 85
 - BOSS_4, 86
 - CREEPER_1, 86
 - CREEPER_2, 86
 - CREEPER_3, 86
 - CREEPER_4, 86
 - EXPLOSION_1, 86
 - EXPLOSION_2, 87
 - EXPLOSION_3, 87
 - INVADER_11, 87
 - INVADER_12, 87
 - INVADER_13, 87
 - INVADER_14, 87
 - INVADER_21, 88
 - INVADER_22, 88
 - INVADER_23, 88
 - INVADER_24, 88
 - INVADER_31, 88
 - INVADER_32, 88
 - INVADER_33, 89
 - INVADER_34, 89
 - INVADER_41, 89
 - INVADER_42, 89
 - INVADER_43, 89
 - INVADER_44, 89
 - SHIP_1, 90
 - SHIP_2, 90
 - SHIP_3, 90
 - SHIP_4, 90
 - TOUGH_1, 90
 - TOUGH_2, 90
 - TOUGH_3, 91
 - TOUGH_4, 91
 - Textures, 81
 - UFO_1, 91
 - UFO_2, 91
 - UFO_3, 91
 - UFO_4, 91
- UFO_1
 - Textures, 91
- UFO_2
 - Textures, 91
- UFO_3
 - Textures, 91
- UFO_4
 - Textures, 91
- update
 - Explosion, 28
 - Explosions, 30
 - InvaderFormation, 43
 - Load, 53
 - Menu, 57
 - MenuIG, 60

- Save, 68
- Spaceship, 74
- update_objects
 - Game, 13
- updateStars
 - Stars, 78
- willHurt
 - InvaderLaser, 48