

Síkidomok

Készítette Doxygen 1.8.17

1. Névtérmutató	1
1.1. Névtérlista	1
2. Hierarchikus mutató	3
2.1. Osztályhierarchia	3
3. Osztálymutató	5
3.1. Osztálylista	5
4. Fájlmutató	7
4.1. Fájllista	7
5. Névterek dokumentációja	9
5.1. anonymous_namespace{console.cpp} névtér-referencia	9
5.1.1. Részletes leírás	9
5.2. anonymous_namespace{game_asteroids.cpp} névtér-referencia	9
5.2.1. Függvények dokumentációja	9
5.2.1.1. randd()	9
5.3. anonymous_namespace{main_assignment.cpp} névtér-referencia	9
5.3.1. Függvények dokumentációja	10
5.3.1.1. outside_of_unit_circle()	10
5.4. anonymous_namespace{screen.cpp} névtér-referencia	10
5.4.1. Enumerációk dokumentációja	10
5.4.1.1. Block	10
5.4.2. Függvények dokumentációja	10
5.4.2.1. getblock()	10
5.5. gtest_lite névtér-referencia	11
5.5.1. Részletes leírás	11
5.5.2. Függvények dokumentációja	12
5.5.2.1. almostEQ()	12
5.5.2.2. eq()	12
5.5.2.3. eqstr()	12
5.5.2.4. eqstrcase()	12
5.5.2.5. EXPECT_() [1/2]	12
5.5.2.6. EXPECT_() [2/2]	13
5.5.2.7. EXPECTSTR()	13
5.5.2.8. ge()	13
5.5.2.9. gt()	13
5.5.2.10. le()	14
5.5.2.11. lt()	14
5.5.2.12. ne()	14
5.5.2.13. nestr()	14
5.5.3. Változók dokumentációja	14
5.5.3.1. test	14

6. Osztályok dokumentációja	15
6.1. <code>_Is_Types< F, T ></code> struktúrasablon-referencia	15
6.1.1. Részletes leírás	16
6.1.2. Tagfüggvények dokumentációja	16
6.1.2.1. <code>f()</code> [1/2]	16
6.1.2.2. <code>f()</code> [2/2]	16
6.1.3. Adattagok dokumentációja	16
6.1.3.1. <code>convertable</code>	16
6.2. <code>GameAsteroids::Actor</code> struktúrareferencia	17
6.2.1. Részletes leírás	18
6.2.2. Konstruktorkok és destruktorkok dokumentációja	18
6.2.2.1. <code>Actor()</code> [1/2]	18
6.2.2.2. <code>Actor()</code> [2/2]	18
6.2.3. Tagfüggvények dokumentációja	18
6.2.3.1. <code>update()</code>	18
6.2.4. Adattagok dokumentációja	19
6.2.4.1. <code>pos</code>	19
6.2.4.2. <code>rot</code>	19
6.2.4.3. <code>size</code>	19
6.2.4.4. <code>speed</code>	19
6.3. <code>Circle</code> osztályreferencia	20
6.3.1. Részletes leírás	22
6.3.2. Konstruktorkok és destruktorkok dokumentációja	22
6.3.2.1. <code>Circle()</code> [1/3]	22
6.3.2.2. <code>Circle()</code> [2/3]	22
6.3.2.3. <code>Circle()</code> [3/3]	22
6.3.3. Tagfüggvények dokumentációja	22
6.3.3.1. <code>get_center()</code>	22
6.3.3.2. <code>has_point()</code>	23
6.3.3.3. <code>intersects_with()</code>	23
6.3.3.4. <code>print_to()</code>	23
6.3.3.5. <code>read_from()</code>	23
6.3.4. Adattagok dokumentációja	23
6.3.4.1. <code>center</code>	24
6.3.4.2. <code>radius</code>	24
6.4. <code>Console</code> osztályreferencia	24
6.4.1. Részletes leírás	25
6.4.2. Konstruktorkok és destruktorkok dokumentációja	26
6.4.2.1. <code>Console()</code> [1/2]	26
6.4.2.2. <code>Console()</code> [2/2]	26
6.4.2.3. <code>~Console()</code>	26
6.4.3. Tagfüggvények dokumentációja	26

6.4.3.1.	<code>clrscr()</code>	26
6.4.3.2.	<code>con()</code>	26
6.4.3.3.	<code>getch()</code>	27
6.4.3.4.	<code>getsize()</code>	27
6.4.3.5.	<code>gotoxy()</code>	27
6.4.3.6.	<code>hMeter()</code>	27
6.4.3.7.	<code>kbhit()</code>	28
6.4.3.8.	<code>operator=()</code>	28
6.4.3.9.	<code>trCode()</code>	28
6.4.4.	Adattagok dokumentációja	29
6.4.4.1.	<code>KEY_DOWN</code>	29
6.4.4.2.	<code>KEY_HOME</code>	29
6.4.4.3.	<code>KEY_LEFT</code>	29
6.4.4.4.	<code>KEY_RIGHT</code>	29
6.4.4.5.	<code>KEY_UP</code>	29
6.5.	<code>DynArray< T ></code> osztálysablon-referencia	30
6.5.1.	Részletes leírás	32
6.5.2.	Konstruktorok és destruktorok dokumentációja	32
6.5.2.1.	<code>DynArray()</code> [1/2]	32
6.5.2.2.	<code>DynArray()</code> [2/2]	32
6.5.2.3.	<code>~DynArray()</code>	32
6.5.3.	Tagfüggvények dokumentációja	33
6.5.3.1.	<code>append()</code>	33
6.5.3.2.	<code>begin()</code>	33
6.5.3.3.	<code>end()</code>	33
6.5.3.4.	<code>operator=()</code>	33
6.5.4.	Adattagok dokumentációja	33
6.5.4.1.	<code>data</code>	33
6.5.4.2.	<code>DEFAULT_SIZE</code>	34
6.5.4.3.	<code>length</code>	34
6.5.4.4.	<code>size</code>	34
6.6.	<code>Game</code> osztályreferencia	34
6.6.1.	Részletes leírás	35
6.6.2.	Konstruktorok és destruktorok dokumentációja	35
6.6.2.1.	<code>~Game()</code>	35
6.6.3.	Tagfüggvények dokumentációja	35
6.6.3.1.	<code>input()</code>	36
6.6.3.2.	<code>play()</code>	36
6.6.3.3.	<code>update()</code>	36
6.7.	<code>GameAsteroids</code> osztályreferencia	37
6.7.1.	Részletes leírás	39
6.7.2.	Konstruktorok és destruktorok dokumentációja	39

6.7.2.1.	GameAsteroids()	39
6.7.3.	Tagfüggvények dokumentációja	39
6.7.3.1.	input()	39
6.7.3.2.	spawn_random_asteroid()	40
6.7.3.3.	update()	40
6.7.4.	Adattagok dokumentációja	41
6.7.4.1.	acceleration	41
6.7.4.2.	asteroids	41
6.7.4.3.	player	41
6.7.4.4.	seconds_since_asteroid	41
6.7.4.5.	seconds_until_asteroid	41
6.8.	GameFlappyBird osztályreferencia	42
6.8.1.	Részletes leírás	44
6.8.2.	Konstruktorok és destruktorok dokumentációja	44
6.8.2.1.	GameFlappyBird()	44
6.8.3.	Tagfüggvények dokumentációja	44
6.8.3.1.	input()	44
6.8.3.2.	randomize_spike_height()	44
6.8.3.3.	update()	44
6.8.4.	Adattagok dokumentációja	45
6.8.4.1.	bird_pos	45
6.8.4.2.	bird_velocity	45
6.8.4.3.	gravity	45
6.8.4.4.	jump_speed	45
6.8.4.5.	lower_wall	45
6.8.4.6.	spike_x	46
6.8.4.7.	spike_x_speed	46
6.8.4.8.	spike_y_offset	46
6.8.4.9.	upper_wall	46
6.9.	GameSnake osztályreferencia	46
6.9.1.	Részletes leírás	49
6.9.2.	Konstruktorok és destruktorok dokumentációja	49
6.9.2.1.	GameSnake()	49
6.9.3.	Tagfüggvények dokumentációja	49
6.9.3.1.	input()	49
6.9.3.2.	update()	49
6.9.4.	Adattagok dokumentációja	50
6.9.4.1.	block_size	50
6.9.4.2.	delta_since_last_jump	50
6.9.4.3.	forward	50
6.9.4.4.	seconds_till_snake_jumps	50
6.9.4.5.	shape_parser	50

6.9.4.6. SNAKE_LENGTH	50
6.9.4.7. snake_pos	51
6.10. ShapeParser::Iterator osztályreferencia	51
6.10.1. Részletes leírás	52
6.10.2. Típusdefiníció-tagok dokumentációja	53
6.10.2.1. value_type	53
6.10.3. Konstruktork és destruktorok dokumentációja	53
6.10.3.1. Iterator()	53
6.10.4. Tagfüggvények dokumentációja	53
6.10.4.1. operator*()	53
6.11. DynArray< T >::Iterator osztályreferencia	54
6.11.1. Részletes leírás	55
6.11.2. Típusdefiníció-tagok dokumentációja	55
6.11.2.1. difference_type	55
6.11.2.2. iterator_category	55
6.11.2.3. pointer	56
6.11.2.4. reference	56
6.11.2.5. value_type	56
6.11.3. Konstruktork és destruktorok dokumentációja	56
6.11.3.1. Iterator()	56
6.11.4. Tagfüggvények dokumentációja	56
6.11.4.1. operator"!=()	56
6.11.4.2. operator*()	56
6.11.4.3. operator++()	57
6.11.5. Adattagok dokumentációja	57
6.11.5.1. current	57
6.11.5.2. end	57
6.12. Console::keyCodes struktúrareferencia	57
6.12.1. Részletes leírás	58
6.12.2. Adattagok dokumentációja	58
6.12.2.1. code	58
6.12.2.2. key	58
6.13. gtest_lite::ostreamRedir osztályreferencia	58
6.13.1. Részletes leírás	59
6.13.2. Konstruktork és destruktorok dokumentációja	59
6.13.2.1. ostreamRedir()	59
6.13.2.2. ~ostreamRedir()	59
6.13.3. Adattagok dokumentációja	59
6.13.3.1. save	59
6.13.3.2. src	60
6.14. Polygon osztályreferencia	60
6.14.1. Részletes leírás	62

6.14.2. Konstruktorok és destruktorok dokumentációja	62
6.14.2.1. Polygon() [1/2]	62
6.14.2.2. Polygon() [2/2]	62
6.14.3. Tagfüggvények dokumentációja	62
6.14.3.1. closest_edge_to()	63
6.14.3.2. has_point()	63
6.14.3.3. intersects_with()	63
6.14.3.4. print_to()	63
6.14.3.5. read_from()	63
6.14.4. Adattagok dokumentációja	64
6.14.4.1. center	64
6.14.4.2. vertex	64
6.14.4.3. vertex_count	64
6.15. Screen osztályreferencia	64
6.15.1. Részletes leírás	66
6.15.2. Konstruktorok és destruktorok dokumentációja	66
6.15.2.1. Screen() [1/2]	66
6.15.2.2. Screen() [2/2]	66
6.15.2.3. ~Screen()	67
6.15.3. Tagfüggvények dokumentációja	67
6.15.3.1. clear()	67
6.15.3.2. draw_shape()	67
6.15.3.3. idx() [1/2]	67
6.15.3.4. idx() [2/2]	67
6.15.3.5. operator=()	67
6.15.3.6. render()	68
6.15.4. Adattagok dokumentációja	68
6.15.4.1. data	68
6.15.4.2. height	68
6.15.4.3. size	68
6.15.4.4. width	68
6.16. Segment struktúráreferencia	69
6.16.1. Részletes leírás	70
6.16.2. Konstruktorok és destruktorok dokumentációja	70
6.16.2.1. Segment()	70
6.16.3. Tagfüggvények dokumentációja	70
6.16.3.1. closest_point_to()	70
6.16.3.2. is_point_to_the_left()	70
6.16.4. Adattagok dokumentációja	70
6.16.4.1. a	71
6.16.4.2. b	71
6.17. Shape osztályreferencia	71

6.17.1. Részletes leírás	72
6.17.2. Konstruktork és destruktorok dokumentációja	72
6.17.2.1. ~Shape()	72
6.17.3. Tagfüggvények dokumentációja	72
6.17.3.1. has_point()	73
6.17.3.2. intersects_with()	73
6.17.3.3. print_to()	73
6.17.3.4. read_from()	73
6.18. ShapeParser osztályreferencia	74
6.18.1. Részletes leírás	75
6.18.2. Típusdefiníció-tagok dokumentációja	75
6.18.2.1. SuperIt	75
6.18.3. Konstruktork és destruktorok dokumentációja	75
6.18.3.1. ShapeParser() [1/2]	75
6.18.3.2. ShapeParser() [2/2]	75
6.18.3.3. ~ShapeParser()	76
6.18.4. Tagfüggvények dokumentációja	76
6.18.4.1. begin()	76
6.18.4.2. end()	76
6.18.4.3. operator=()	76
6.18.5. Adattagok dokumentációja	76
6.18.5.1. array	76
6.19. gtest_lite::Test struktúráreferencia	77
6.19.1. Részletes leírás	78
6.19.2. Konstruktork és destruktorok dokumentációja	78
6.19.2.1. Test() [1/2]	78
6.19.2.2. Test() [2/2]	78
6.19.2.3. ~Test()	79
6.19.3. Tagfüggvények dokumentációja	79
6.19.3.1. astatus()	79
6.19.3.2. begin()	79
6.19.3.3. end()	79
6.19.3.4. expect()	79
6.19.3.5. fail()	80
6.19.3.6. getTest()	80
6.19.3.7. operator=()	80
6.19.4. Adattagok dokumentációja	80
6.19.4.1. ablocks	80
6.19.4.2. failed	80
6.19.4.3. name	80
6.19.4.4. null	81
6.19.4.5. os	81

6.19.4.6. status	81
6.19.4.7. sum	81
6.19.4.8. tmp	81
6.20. Vector struktúráreferencia	82
6.20.1. Részletes leírás	83
6.20.2. Konstruktork és destruktork dokumentációja	83
6.20.2.1. Vector() [1/2]	83
6.20.2.2. Vector() [2/2]	83
6.20.3. Tagfüggvények dokumentációja	83
6.20.3.1. angle_to()	84
6.20.3.2. distance_squared_to()	84
6.20.3.3. distance_to()	84
6.20.3.4. operator*() [1/2]	84
6.20.3.5. operator*() [2/2]	84
6.20.3.6. operator+()	84
6.20.3.7. operator+=()	85
6.20.3.8. operator-()	85
6.20.3.9. polar()	85
6.20.3.10. rotate()	85
6.20.3.11. rotate_around()	85
6.20.4. Adattagok dokumentációja	86
6.20.4.1. DOWN	86
6.20.4.2. LEFT	86
6.20.4.3. RIGHT	86
6.20.4.4. UP	86
6.20.4.5. x	86
6.20.4.6. y	86
7. Fájlok dokumentációja	87
7.1. console.cpp fájlreferencia	87
7.1.1. Részletes leírás	88
7.1.2. Makródefiníciók dokumentációja	88
7.1.2.1. C	88
7.2. console.h fájlreferencia	88
7.2.1. Részletes leírás	89
7.3. dynarray.hpp fájlreferencia	89
7.3.1. Részletes leírás	90
7.4. game.cpp fájlreferencia	90
7.4.1. Részletes leírás	90
7.4.2. Változók dokumentációja	90
7.4.2.1. max_delta	91
7.5. game.h fájlreferencia	91

7.5.1. Részletes leírás	92
7.6. game_asteroids.cpp fájlreferencia	92
7.6.1. Részletes leírás	92
7.7. game_asteroids.h fájlreferencia	93
7.7.1. Részletes leírás	94
7.8. game_flappy_bird.cpp fájlreferencia	94
7.8.1. Részletes leírás	94
7.9. game_flappy_bird.h fájlreferencia	94
7.9.1. Részletes leírás	95
7.10. game_snake.cpp fájlreferencia	96
7.10.1. Részletes leírás	96
7.11. game_snake.h fájlreferencia	96
7.11.1. Részletes leírás	97
7.12. gtest_lite.h fájlreferencia	98
7.12.1. Részletes leírás	101
7.12.2. Makródefiníciók dokumentációja	101
7.12.2.1. ADD_FAILURE	101
7.12.2.2. ASSERT_	102
7.12.2.3. ASSERT_EQ	102
7.12.2.4. ASSERT_NO_THROW [1/2]	102
7.12.2.5. ASSERT_NO_THROW [2/2]	102
7.12.2.6. ASSERTTHROW	102
7.12.2.7. CREATE_Has_	103
7.12.2.8. CREATE_Has_fn_	103
7.12.2.9. END	103
7.12.2.10.ENDM	103
7.12.2.11.ENDMsg	103
7.12.2.12.EXPECT_ANY_THROW	103
7.12.2.13.EXPECT_DOUBLE_EQ	103
7.12.2.14.EXPECT_ENVCASEEQ	104
7.12.2.15.EXPECT_ENVEQ	104
7.12.2.16.EXPECT_EQ	104
7.12.2.17.EXPECT_FALSE	104
7.12.2.18.EXPECT_FLOAT_EQ	104
7.12.2.19.EXPECT_GE	104
7.12.2.20.EXPECT_GT	105
7.12.2.21.EXPECT_LE	105
7.12.2.22.EXPECT_LT	105
7.12.2.23.EXPECT_NE	105
7.12.2.24.EXPECT_NO_THROW	105
7.12.2.25.EXPECT_STRCASEEQ	105
7.12.2.26.EXPECT_STRCASENE	105

7.12.2.27.EXPECT_STREQ	106
7.12.2.28.EXPECT_STRNE	106
7.12.2.29.EXPECT_THROW	106
7.12.2.30.EXPECT_THROW_THROW	106
7.12.2.31.EXPECT_TRUE	106
7.12.2.32.EXPECTTHROW	106
7.12.2.33.Nem célszerű közvetlenül használni, vagy módosítani	107
7.12.2.34.FAIL	107
7.12.2.35.GTEND	107
7.12.2.36.GTINIT	107
7.12.2.37.SUCCEED	107
7.12.2.38.TEST	107
7.12.3. Függvények dokumentációja	107
7.12.3.1. hasMember()	107
7.13. main.cpp fájlreferencia	108
7.13.1. Részletes leírás	108
7.13.2. Függvények dokumentációja	108
7.13.2.1. main()	108
7.14. main_assignment.cpp fájlreferencia	108
7.14.1. Részletes leírás	109
7.14.2. Függvények dokumentációja	109
7.14.2.1. main_assignment()	109
7.15. main_game.cpp fájlreferencia	109
7.15.1. Részletes leírás	110
7.15.2. Függvények dokumentációja	110
7.15.2.1. main_game()	110
7.16. main_test.cpp fájlreferencia	110
7.16.1. Részletes leírás	111
7.16.2. Függvények dokumentációja	111
7.16.2.1. main_test()	111
7.17. mains.h fájlreferencia	111
7.17.1. Részletes leírás	112
7.17.2. Függvények dokumentációja	112
7.17.2.1. main_assignment()	112
7.17.2.2. main_game()	112
7.17.2.3. main_test()	112
7.18. memtrace.cpp fájlreferencia	113
7.19. memtrace.h fájlreferencia	113
7.20. screen.cpp fájlreferencia	113
7.20.1. Részletes leírás	114
7.21. screen.h fájlreferencia	114
7.21.1. Részletes leírás	115

7.21.2. Függvények dokumentációja	116
7.21.2.1. operator<<()	116
7.22. segment.cpp fájlreferencia	116
7.22.1. Részletes leírás	116
7.23. segment.h fájlreferencia	116
7.23.1. Részletes leírás	117
7.23.2. Függvények dokumentációja	117
7.23.2.1. operator<<()	118
7.24. shape.h fájlreferencia	118
7.24.1. Függvények dokumentációja	119
7.24.1.1. operator<<()	119
7.24.1.2. operator>>()	119
7.25. shape_circle.h fájlreferencia	119
7.25.1. Részletes leírás	120
7.26. shape_parser.cpp fájlreferencia	121
7.26.1. Részletes leírás	121
7.27. shape_parser.h fájlreferencia	121
7.27.1. Részletes leírás	122
7.28. shape_polygon.cpp fájlreferencia	122
7.28.1. Részletes leírás	123
7.29. shape_polygon.h fájlreferencia	123
7.29.1. Részletes leírás	124
7.30. shapes.h fájlreferencia	125
7.30.1. Részletes leírás	125
7.31. snake_level.txt fájlreferencia	125
7.32. vector.cpp fájlreferencia	125
7.32.1. Részletes leírás	126
7.33. vector.h fájlreferencia	126
7.33.1. Részletes leírás	127
7.33.2. Függvények dokumentációja	127
7.33.2.1. operator<<()	127
7.33.2.2. operator>>()	127
7.34. vectormath.h fájlreferencia	127
7.34.1. Részletes leírás	128
7.34.2. Változók dokumentációja	128
7.34.2.1. MATH_PI	129

1. fejezet

Névtérmutató

1.1. Névtérlista

Az összes névtér listája rövid leírásokkal:

anonymous_namespace{console.cpp}	9
anonymous_namespace{game_asteroids.cpp}	9
anonymous_namespace{main_assignment.cpp}	9
anonymous_namespace{screen.cpp}	10
gtest_lite	
Gtest_lite: a keretrendszer függvényinek és objektumainak névtére	11

2. fejezet

Hierarchikus mutató

2.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

_Is_Types< F, T >	15
GameAsteroids::Actor	17
Console	24
DynArray< T >	30
DynArray< GameAsteroids::Actor >	30
DynArray< Shape * >	30
Game	34
GameAsteroids	37
GameFlappyBird	42
GameSnake	46
DynArray< T >::Iterator	54
ShapeParser::Iterator	51
Console::keyCodes	57
gtest_lite::ostreamRedir	58
Screen	64
Segment	69
Shape	71
Circle	20
Polygon	60
ShapeParser	74
gtest_lite::Test	77
Vector	82

3. fejezet

Osztálymutató

3.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

_Is_Types< F, T >	
Segédsablon típuskonverzió futás közbeni ellenőrzésére	15
GameAsteroids::Actor	17
Circle	20
Console	24
DynArray< T >	30
Game	34
GameAsteroids	37
GameFlappyBird	42
GameSnake	46
ShapeParser::Iterator	51
DynArray< T >::Iterator	54
Console::keyCodes	
Segédtypus a kódváltáshoz	57
gtest_lite::ostreamRedir	58
Polygon	60
Screen	64
Segment	
Egy szakaszt, vagy egyenest reprezentál	69
Shape	
Absztrakt síkidom őssztály	71
ShapeParser	74
gtest_lite::Test	77
Vector	82

4. fejezet

Fájlmutató

4.1. Fájllista

Az összes fájl listája rövid leírásokkal:

console.cpp	87
console.h	88
dynarray.hpp	89
game.cpp	90
game.h	91
game_asteroids.cpp	92
game_asteroids.h	93
game_flappy_bird.cpp	94
game_flappy_bird.h	94
game_snake.cpp	96
game_snake.h	96
gtest_lite.h	98
main.cpp	108
main_assignment.cpp	108
main_game.cpp	109
main_test.cpp	110
mains.h	111
memtrace.cpp	113
memtrace.h	113
screen.cpp	113
screen.h	114
segment.cpp	116
segment.h	116
shape.h	118
shape_circle.h	119
shape_parser.cpp	121
shape_parser.h	121
shape_polygon.cpp	122
shape_polygon.h	123
shapes.h	125
vector.cpp	125
vector.h	126
vectormath.h	127

5. fejezet

Névterek dokumentációja

5.1. anonymous_namespace{console.cpp} névtér-referencia

5.1.1. Részletes leírás

Noname névtér. Csak ebből a fájlból érhető el

5.2. anonymous_namespace{game_asteroids.cpp} névtér-referencia

Függvények

- double `randd` (double range)

5.2.1. Függvények dokumentációja

5.2.1.1. `randd()`

```
double anonymous_namespace{game_asteroids.cpp}::randd (  
    double range )
```

5.3. anonymous_namespace{main_assignment.cpp} névtér-referencia

Függvények

- bool `outside_of_unit_circle` (const `Shape` &s)

5.3.1. Függvények dokumentációja

5.3.1.1. outside_of_unit_circle()

```
bool anonymous_namespace{main_assignment.cpp}::outside_of_unit_circle (
    const Shape & s )
```

5.4. anonymous_namespace{screen.cpp} névtér-referencia

Enumerációk

- enum Block { BLOCK_EMPTY, BLOCK_DOWN, BLOCK_UP, BLOCK_FULL }

Függvények

- const char * getblock (Block b)
ad egy dobozt: szököz, alsó, felső, vagy teljes

5.4.1. Enumerációk dokumentációja

5.4.1.1. Block

```
enum anonymous_namespace{screen.cpp}::Block
```

Enumeráció-értékek

BLOCK_EMPTY	
BLOCK_DOWN	
BLOCK_UP	
BLOCK_FULL	

5.4.2. Függvények dokumentációja

5.4.2.1. getblock()

```
const char* anonymous_namespace{screen.cpp}::getblock (
    Block b )
```


ad egy dobozt: szóköz, alsó, felső, vagy teljes

5.5. gtest_lite névtér-referencia

[gtest_lite](#): a keretrendszer függvényinek és objektumainak névtére

Osztályok

- class [ostreamRedir](#)
- struct [Test](#)

Függvények

- `template<typename T1 , typename T2 >`
`std::ostream & EXPECT_ (T1 exp, T2 act, bool(*pred)(T1, T1), const char *file, int line, const char *expr,`
`const char *lhs="elvart", const char *rhs="aktual")`
általános sablon a várt értékhez.
- `template<typename T1 , typename T2 >`
`std::ostream & EXPECT_ (T1 *exp, T2 *act, bool(*pred)(T1 *, T1 *), const char *file, int line, const char`
`*expr, const char *lhs="elvart", const char *rhs="aktual")`
pointerre specializált sablon a várt értékhez.
- `std::ostream & EXPECTSTR (const char *exp, const char *act, bool(*pred)(const char *, const char *), const`
`char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")`
- `template<typename T >`
`bool eq (T a, T b)`
- `bool eqstr (const char *a, const char *b)`
- `bool eqstrcase (const char *a, const char *b)`
- `template<typename T >`
`bool ne (T a, T b)`
- `bool nestr (const char *a, const char *b)`
- `template<typename T >`
`bool le (T a, T b)`
- `template<typename T >`
`bool lt (T a, T b)`
- `template<typename T >`
`bool ge (T a, T b)`
- `template<typename T >`
`bool gt (T a, T b)`
- `template<typename T >`
`bool almostEQ (T a, T b)`

Változók

- static [Test](#) & [test](#) = [Test::getTest](#)()

5.5.1. Részletes leírás

[gtest_lite](#): a keretrendszer függvényinek és objektumainak névtére

5.5.2. Függvények dokumentációja

5.5.2.1. almostEQ()

```
template<typename T >
bool gtest_lite::almostEQ (
    T a,
    T b )
```

Segédsablon valós számok összehasonlításához Nem bombabiztos, de nekünk most jó lesz Elméleti hátér:
<http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm>

5.5.2.2. eq()

```
template<typename T >
bool gtest_lite::eq (
    T a,
    T b )
```

segéd sablonok a relációkhoz. azért nem STL (algorithm), mert csak a függvény lehet, hogy menjen a deduckció

5.5.2.3. eqstr()

```
bool gtest_lite::eqstr (
    const char * a,
    const char * b ) [inline]
```

5.5.2.4. eqstrcase()

```
bool gtest_lite::eqstrcase (
    const char * a,
    const char * b ) [inline]
```

5.5.2.5. EXPECT_() [1/2]

```
template<typename T1 , typename T2 >
std::ostream& gtest_lite::EXPECT_ (
    T1 * exp,
    T2 * act,
    bool(*) (T1 *, T1 *) pred,
    const char * file,
    int line,
    const char * expr,
    const char * lhs = "elvart",
    const char * rhs = "aktual" )
```

pointerre specializált sablon a várt értékhez.

5.5.2.6. EXPECT_() [2/2]

```
template<typename T1 , typename T2 >
std::ostream& gtest_lite::EXPECT_ (
    T1 exp,
    T2 act,
    bool(*) (T1, T1) pred,
    const char * file,
    int line,
    const char * expr,
    const char * lhs = "elvart",
    const char * rhs = "aktual" )
```

általános sablon a várt értékhez.

5.5.2.7. EXPECTSTR()

```
std::ostream& gtest_lite::EXPECTSTR (
    const char * exp,
    const char * act,
    bool(*) (const char *, const char *) pred,
    const char * file,
    int line,
    const char * expr,
    const char * lhs = "elvart",
    const char * rhs = "aktual" ) [inline]
```

stringek összehasonlításához. azért nem spec. mert a sima EQ-ra másként kell működnie.

5.5.2.8. ge()

```
template<typename T >
bool gtest_lite::ge (
    T a,
    T b )
```

5.5.2.9. gt()

```
template<typename T >
bool gtest_lite::gt (
    T a,
    T b )
```

5.5.2.10. le()

```
template<typename T >
bool gtest_lite::le (
    T a,
    T b )
```

5.5.2.11. lt()

```
template<typename T >
bool gtest_lite::lt (
    T a,
    T b )
```

5.5.2.12. ne()

```
template<typename T >
bool gtest_lite::ne (
    T a,
    T b )
```

5.5.2.13. nestr()

```
bool gtest_lite::nestr (
    const char * a,
    const char * b ) [inline]
```

5.5.3. Változók dokumentációja

5.5.3.1. test

```
Test& gtest_lite::test = Test::getTest() [static]
```

A statikus referencia minden fordítási egységben keletkezik, de mindegyik egyetlen példányra fog hivatkozni a singleton minta miatt

6. fejezet

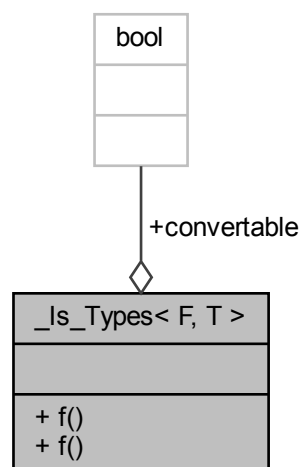
Osztályok dokumentációja

6.1. `_Is_Types< F, T >` struktúrasablon-referencia

Segédsablon típuskonverzió futás közbeni ellenőrzésére.

```
#include <gtest_lite.h>
```

Az `_Is_Types< F, T >` osztály együttműködési diagramja:



Statikus publikus tagfüggvények

- `template<typename D >`
`static char(& f (D))[1]`
- `template<typename D >`
`static char(& f (...))[2]`

Statikus publikus attribútumok

- static const bool `convertable` = sizeof(`f<T>(F())`) == 1

6.1.1. Részletes leírás

```
template<typename F, typename T>
struct _Is_Types< F, T >
```

Segédsablon típuskonverzió futás közbeni ellenőrzésére.

6.1.2. Tagfüggvények dokumentációja

6.1.2.1. `f()` [1/2]

```
template<typename F , typename T >
template<typename D >
static char(& _Is_Types< F, T >::f (
    ... )) [2] [static]
```

6.1.2.2. `f()` [2/2]

```
template<typename F , typename T >
template<typename D >
static char(& _Is_Types< F, T >::f (
    D )) [1] [static]
```

6.1.3. Adattagok dokumentációja

6.1.3.1. `convertable`

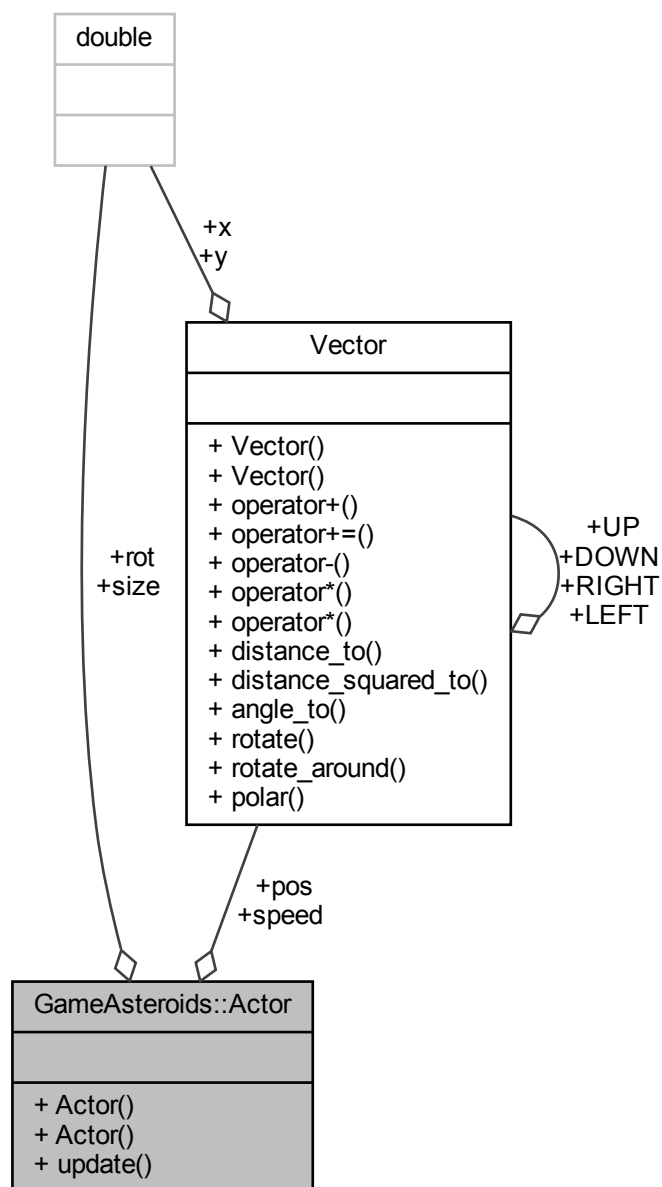
```
template<typename F , typename T >
const bool _Is_Types< F, T >::convertable = sizeof(f<T>(F())) == 1 [static]
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- `gtest_lite.h`

6.2. GameAsteroids::Actor struktúrareferencia

A GameAsteroids::Actor osztály együttműködési diagramja:



Publikus tagfüggvények

- [Actor](#) ()
inicializáció memóriaszeméttel, a DynArray-hoz kell, ahol az elemek úgyis érvénytelenek
- [Actor](#) ([Vector](#) p, [Vector](#) s, double sz)
- void [update](#) (double delta)

Publikus attribútumok

- [Vector pos](#)
- [Vector speed](#)
- double [size](#)
- double [rot](#)

6.2.1. Részletes leírás

itt az enkapszuláció nagyobb szinten történik: a [GameAsteroids](#) szintjén, ezért nem figyel az [Actor](#) magára (publikus tagváltozók, kevés tagfüggvény) inkább nem duzzasztom fel a kódot `get_rot()`, `set_rot()`, stb. boilerplate-tel, minél kevesebb kód, annál kevesebb hibalehetőség hasonlóan felduzzasztaná a kódot egy-egy játékos, illetve aszteroida alosztály létrehozása, és a több, bonyolultabb kód miatt még kevésbé is tudnám érteni, nehezebb lenne követni a (részben virtuális) függvényhívások tömkelegét, pl a kirajzoláshoz vagy a hozzá tartozó [Shape](#) lekérdezéséhez, amikor egyszerűbben meg lehet oldani ezek nélkül

6.2.2. Konstruktorkok és destruktorkok dokumentációja

6.2.2.1. Actor() [1/2]

```
GameAsteroids::Actor::Actor ( ) [inline]
```

inicializáció memóriaszeméttel, a `DynArray`-hoz kell, ahol az elemek úgylis érvénytelenek

6.2.2.2. Actor() [2/2]

```
GameAsteroids::Actor::Actor (
    Vector p,
    Vector s,
    double sz ) [inline]
```

6.2.3. Tagfüggvények dokumentációja

6.2.3.1. update()

```
void GameAsteroids::Actor::update (
    double delta )
```


6.2.4. Adattagok dokumentációja

6.2.4.1. pos

`Vector GameAsteroids::Actor::pos`

6.2.4.2. rot

`double GameAsteroids::Actor::rot`

6.2.4.3. size

`double GameAsteroids::Actor::size`

6.2.4.4. speed

`Vector GameAsteroids::Actor::speed`

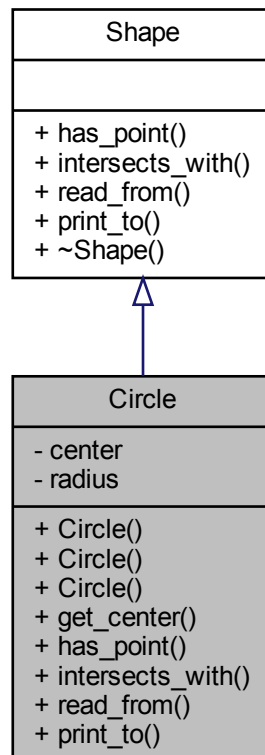
Ez a dokumentáció a struktúráról a következő fájlok alapján készült:

- [game_asteroids.h](#)
- [game_asteroids.cpp](#)

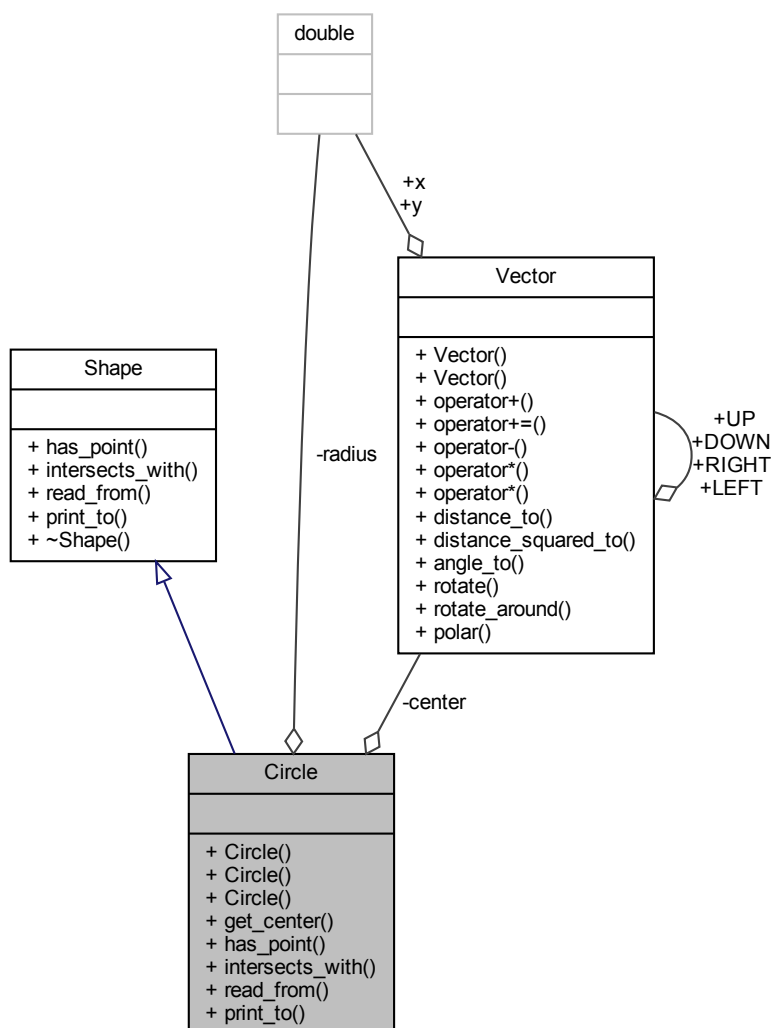
6.3. Circle osztályreferencia

```
#include <shape_circle.h>
```

A Circle osztály származási diagramja:



A Circle osztály együttműködési diagramja:



Publikus tagfüggvények

- **Circle ()**
Default konstruktor csak az adatfolyamból beolvasás céljára, amúgy memóriaszeméttel inicializál.
- **Circle (Vector center, Vector arc_point)**
- **Circle (Vector center, double radius)**
- **Vector get_center () const**
- **bool has_point (Vector p) const**
Visszaadja, hogy a síkidomban benne van-e p pont.
- **bool intersects_with (const Circle &c) const**
Visszaadja, hogy a síkidomnak és c-nek van-e közös pontja (= metszik-e egymást).
- **void read_from (std::istream &is)**
Beolvassa a síkidomot a bemeneti folyamról, 4 double (center x,y és vertex x,y) formájában.
- **void print_to (std::ostream &os) const**
Kiírja a síkidomot a kimeneti folyamra, emberileg olvasható formában, tehát ez nem alkalmazható a read_from-mal közvetlenül perzisztens viselkedés megvalósítására. (A feladat nem is kért ilyet.)

Privát attribútumok

- [Vector](#) *center*
- double *radius*

6.3.1. Részletes leírás

A kör alakzatot megvalósító osztály.

6.3.2. Konstruktorkok és destruktorkok dokumentációja

6.3.2.1. Circle() [1/3]

```
Circle::Circle ( ) [inline]
```

Default konstruktor csak az adatfolyamból beolvasás céljára, amúgy memóriaszeméttel inicializál.

6.3.2.2. Circle() [2/3]

```
Circle::Circle (
    Vector center,
    Vector arc_point ) [inline]
```

6.3.2.3. Circle() [3/3]

```
Circle::Circle (
    Vector center,
    double radius ) [inline]
```

6.3.3. Tagfüggvények dokumentációja

6.3.3.1. get_center()

```
Vector Circle::get_center ( ) const [inline]
```

6.3.3.2. has_point()

```
bool Circle::has_point (
    Vector p ) const [inline], [virtual]
```

Visszaadja, hogy a síkidomban benne van-e p pont.

Megvalósítja a következőket: [Shape](#).

6.3.3.3. intersects_with()

```
bool Circle::intersects_with (
    const Circle & c ) const [inline], [virtual]
```

Visszaadja, hogy a síkidomnak és c-nek van-e közös pontja (= metszik-e egymást).

Megvalósítja a következőket: [Shape](#).

6.3.3.4. print_to()

```
void Circle::print_to (
    std::ostream & os ) const [inline], [virtual]
```

Kírja a síkidomot a kimeneti folyamra, emberileg olvasható formában, tehát ez nem alkalmazható a read_from-mal közvetlenül perzisztens viselkedés megvalósítására. (A feladat nem is kért ilyet.)

Megvalósítja a következőket: [Shape](#).

6.3.3.5. read_from()

```
void Circle::read_from (
    std::istream & is ) [inline], [virtual]
```

Beolvassa a síkidomot a bemeneti folyamról, 4 double (center x,y és vertex x,y) formájában.

Megvalósítja a következőket: [Shape](#).

6.3.4. Adattagok dokumentációja

6.3.4.1. center

```
Vector Circle::center [private]
```

6.3.4.2. radius

```
double Circle::radius [private]
```

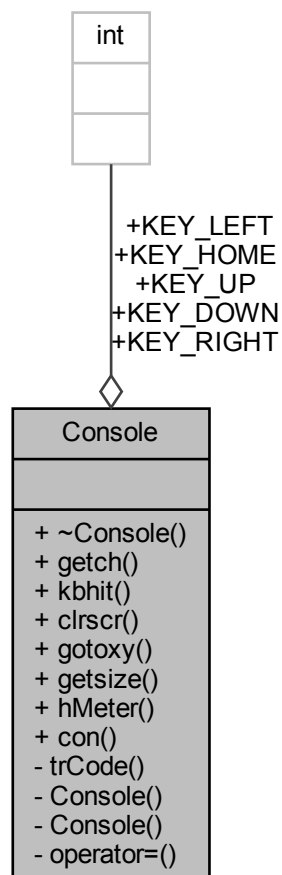
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [shape_circle.h](#)

6.4. Console osztályreferencia

```
#include <console.h>
```

A Console osztály együttműködési diagramja:



Osztályok

- struct `keyCodes`

Segédtypus a kódváltáshoz.

Publikus tagfüggvények

- `~Console ()`

Destruktor.

- int `getch ()`

- bool `kbhit ()`

- void `clrscr ()`

Képernyő törlés.

- void `gotoxy (int x, int y)`

- void `getsize (int &x, int &y)`

Maximum képernyő méret lekérdezés.

- void `hMeter (double value, double max, int width=70, const char *txt=0, int y=0, int x=0)`

Statikus publikus tagfüggvények

- static `Console & con ()`

példányosítás

Statikus publikus attribútumok

- static const int `KEY_DOWN` = 0x102
- static const int `KEY_UP` = 0x103
- static const int `KEY_LEFT` = 0x104
- static const int `KEY_RIGHT` = 0x105
- static const int `KEY_HOME` = 0x106

Privát tagfüggvények

- int `trCode (int code, keyCodes *kt)`

- `Console ()`

csak belülről érhető el

- `Console (const Console &)`

nem engedélyezzük

- `Console & operator= (const Console &)`

ezt sem

6.4.1. Részletes leírás

`Console` ablak kezelését támogató osztály. Singleton minta szerint valósítjuk meg: csak egyetlen példányban létezik. Egy statikus objektummal a singleton minta nem teljesíthető, mert nem lehet kontrollálni az objektum élettartamát. (Később tanulnak róla) A bemutatott megoldásban is csak a keletkezést lehet befolyásolni, a megszűnést nem (de ez már megfelel a mintának).

6.4.2. Konstruktorkok és destruktorkok dokumentációja

6.4.2.1. Console() [1/2]

```
Console::Console ( ) [private]
```

csak belülről érhető el

UNIX/Linux alatt át ki kell kapcsolni az echo-t és a kanonikus módot.

6.4.2.2. Console() [2/2]

```
Console::Console (
    const Console & ) [private]
```

nem engedélyezzük

6.4.2.3. ~Console()

```
Console::~~Console ( )
```

Destruktor.

Elmentett működési módok visszaállítása.

6.4.3. Tagfüggvények dokumentációja

6.4.3.1. clrscr()

```
void Console::clrscr ( )
```

Képernyő törlés.

6.4.3.2. con()

```
static Console& Console::con ( ) [inline], [static]
```

példányosítás

a függvény első futásakor fut el a konstruktora

6.4.3.3. getch()

```
int Console::getch ( )
```

Egy karakter olvasása echo nélkül Ncurses-szerű inputot ad.

Visszatérési érték

olvasott karakter kódja a speciális billentyűk 256 fölé kerülnek

segédmakró

6.4.3.4. getsize()

```
void Console::getsize (
    int & x,
    int & y )
```

Maximum képernyő méret lekérdezés.

Maximum képernyő méret lekérdezés Hiba esetén (legalábbis Windowson) nem módosítja x és y tartalmát!

6.4.3.5. gotoxy()

```
void Console::gotoxy (
    int x,
    int y )
```

Pozicionálás a képernyőn (1,1) a kezdő pozíció a bal felső sarok

Paraméterek

<i>x</i>	- vízszintes pozíció (1..80)
<i>y</i>	- függőleges pozíció (1..24?)

6.4.3.6. hMeter()

```
void Console::hMeter (
    double value,
    double max,
    int width = 70,
    const char * txt = 0,
    int y = 0,
    int x = 0 )
```

Valós érték nagyságának megjelenítése egy vízszintes vonallal

Paraméterek

<i>value</i>	- érték
<i>max</i>	- érték maximuma
<i>width</i>	- megjelenítés szélessége
<i>txt</i>	- vonal előtt kiírandó szöveg. Tartalmazhat printf formátumstringet a value-ra
<i>y</i>	- vonal függőleges kezdő pozíciója
<i>x</i>	- vonal vízszintes kezdő pozíciója

milyen hosszú lesz?

annyit foglalunk és beleírjuk

kiírtuk, nem kell már a buffer

ennyivel csökken a szélesség

két szöglet miatt

kirajzoljuk a vonalat

6.4.3.7. kbhit()

```
bool Console::kbhit ( )
```

Visszatérési érték

hogyan vár-e a bemeneten karakter lekezelésre, a billentyűzet meg lett-e nyomva

hogyan vár-e a bemeneten karakter lekezelésre

6.4.3.8. operator=()

```
Console& Console::operator= (
    const Console & ) [private]
```

ezt sem

6.4.3.9. trCode()

```
int Console::trCode (
    int code,
    keyCodes * kt ) [inline], [private]
```

Segédfüggvény a kódváltáshoz.

Paraméterek

<i>code</i>	- bejövő kód
<i>kt</i>	- kod-újkód párok tömbje

Visszatérési érték

ujkód, ha sikerült a váltás egyébként 0

6.4.4. Adattagok dokumentációja

6.4.4.1. KEY_DOWN

```
const int Console::KEY_DOWN = 0x102 [static]
```

6.4.4.2. KEY_HOME

```
const int Console::KEY_HOME = 0x106 [static]
```

6.4.4.3. KEY_LEFT

```
const int Console::KEY_LEFT = 0x104 [static]
```

6.4.4.4. KEY_RIGHT

```
const int Console::KEY_RIGHT = 0x105 [static]
```

6.4.4.5. KEY_UP

```
const int Console::KEY_UP = 0x103 [static]
```

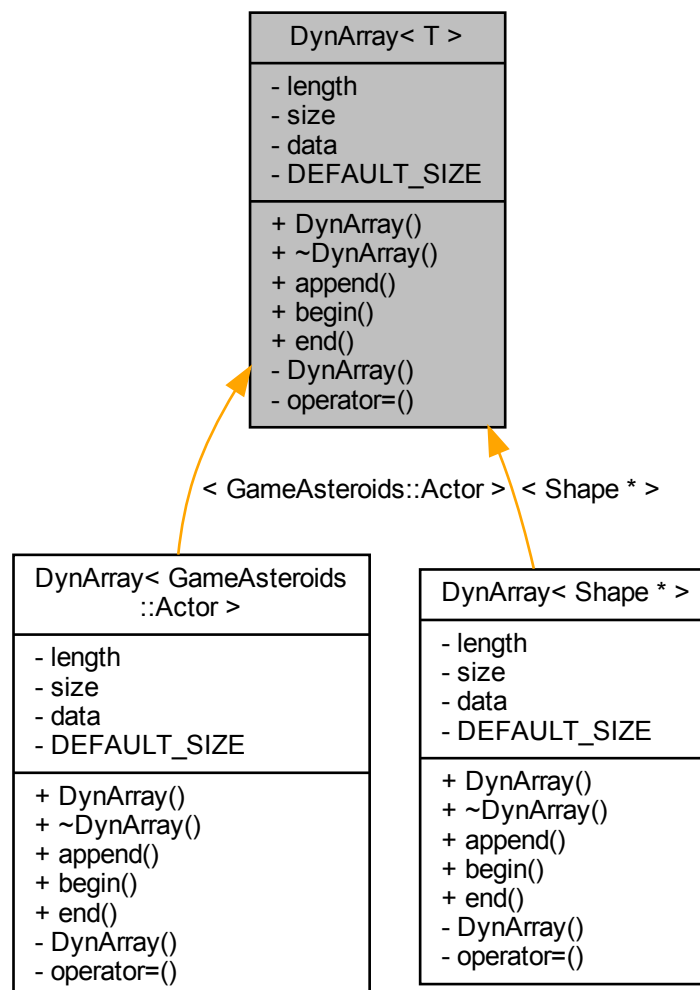
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [console.h](#)
- [console.cpp](#)

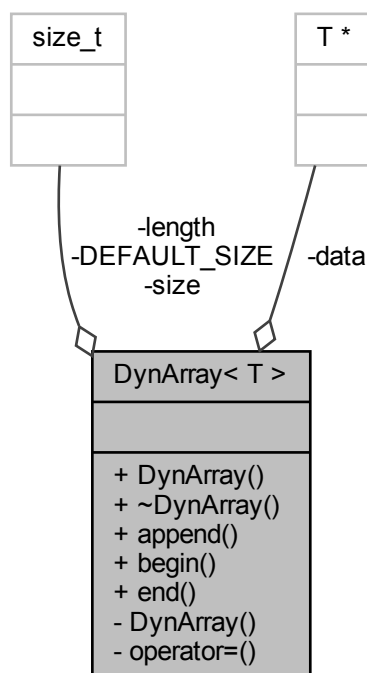
6.5. DynArray< T > osztálysablon-referencia

```
#include <dynarray.hpp>
```

A DynArray< T > osztály származási diagramja:



A DynArray< T > osztály együttműködési diagramja:



Osztályok

- class [Iterator](#)

Publikus tagfüggvények

- [DynArray](#) ()
- [~DynArray](#) ()
- void [append](#) (T appendee)

Hozzáfűzi az elemet a tömb végéhez. Ha túl kicsi, akkor megnyújtja.

- [Iterator begin](#) ()
- [Iterator end](#) ()

Privát tagfüggvények

- [DynArray](#) (const [DynArray](#) &other)
- [DynArray](#) & [operator=](#) (const [DynArray](#) &other)

Privát attribútumok

- `size_t length`
number of valid elements
- `size_t size`
size of data (there may be invalid elements at the end)
- `T * data`

Statikus privát attribútumok

- `constexpr static size_t DEFAULT_SIZE = 4`

6.5.1. Részletes leírás

```
template<typename T>
class DynArray< T >
```

Sablonos dinamikus tömb megvalósítása.

Az elemek csak iterátorral érhetők el, és hozzáfűzni lehet csak, törölni nem. Ha pointer típust tárol dinamikus memóriára, akkor a használó feladata a felszabadítás. Mindig duplázza a tömb hosszát, ha elfogynak a helyek.

6.5.2. Konstruktorkok és destruktorkok dokumentációja

6.5.2.1. DynArray() [1/2]

```
template<typename T >
DynArray< T >::DynArray (
    const DynArray< T > & other ) [private]
```

6.5.2.2. DynArray() [2/2]

```
template<typename T >
DynArray< T >::DynArray ( ) [inline]
```

6.5.2.3. ~DynArray()

```
template<typename T >
DynArray< T >::~~DynArray ( ) [inline]
```

6.5.3. Tagfüggvények dokumentációja

6.5.3.1. append()

```
template<typename T >
void DynArray< T >::append (
    T appendee ) [inline]
```

Hozzáfűzi az elemet a tömb végéhez. Ha túl kicsi, akkor megnyújtja.

6.5.3.2. begin()

```
template<typename T >
Iterator DynArray< T >::begin ( ) [inline]
```

6.5.3.3. end()

```
template<typename T >
Iterator DynArray< T >::end ( ) [inline]
```

6.5.3.4. operator=()

```
template<typename T >
DynArray& DynArray< T >::operator= (
    const DynArray< T > & other ) [private]
```

6.5.4. Adattagok dokumentációja

6.5.4.1. data

```
template<typename T >
T* DynArray< T >::data [private]
```

6.5.4.2. DEFAULT_SIZE

```
template<typename T >
constexpr static size_t DynArray< T >::DEFAULT_SIZE = 4 [static], [constexpr], [private]
```

6.5.4.3. length

```
template<typename T >
size_t DynArray< T >::length [private]
```

number of valid elements

6.5.4.4. size

```
template<typename T >
size_t DynArray< T >::size [private]
```

size of data (there may be invalid elements at the end)

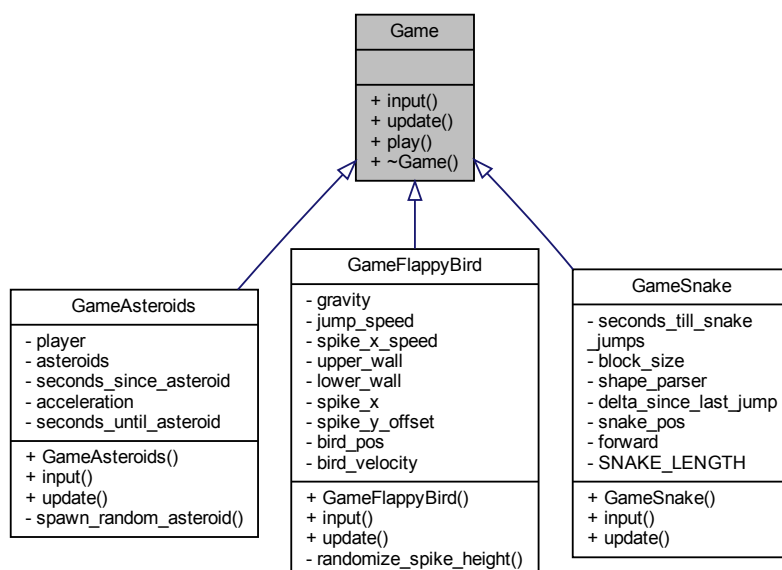
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [dynarray.hpp](#)

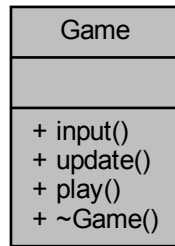
6.6. Game osztályreferencia

```
#include <game.h>
```

A Game osztály származási diagramja:



A Game osztály együttműködési diagramja:



Publikus tagfüggvények

- virtual void `input` (int code)=0
 - virtual bool `update` (double delta, `Screen` &screen)=0
 - bool `play` (`Console` &con, `Screen` &screen, double timeout)
- Minden játék főciklusa, általánosítva.*
- virtual `~Game` ()

6.6.1. Részletes leírás

Absztrakt játék alaposztály, a közösen kezelhetőség céljából.

Meghívódik az `update` függvénye, ha frissíteni lehet a képet, és az `input` függvénye, ha bemenet érkezett. Ezeket kell felülírni, a `play`-t nem.

6.6.2. Konstruktorkok és destruktorkok dokumentációja

6.6.2.1. `~Game()`

```
virtual Game::~~Game ( ) [inline], [virtual]
```

6.6.3. Tagfüggvények dokumentációja

6.6.3.1. input()

```
virtual void Game::input (
    int code ) [pure virtual]
```

Megvalósítják a következők: [GameAsteroids](#), [GameSnake](#) és [GameFlappyBird](#).

6.6.3.2. play()

```
bool Game::play (
    Console & con,
    Screen & screen,
    double timeout )
```

Minden játék főciklusa, általánosítva.

Visszatérési érték

igaz, ha az idő lejárt, hamis, ha a játékos meghalt, vagy kilépett

6.6.3.3. update()

```
virtual bool Game::update (
    double delta,
    Screen & screen ) [pure virtual]
```

Paraméterek

<i>delta</i>	az eltelt idő, másodpercben
<i>screen</i>	amire rajzolni kell

Visszatérési érték

hogyan vége van-e a játéknak az update miatt

Megvalósítják a következők: [GameAsteroids](#), [GameSnake](#) és [GameFlappyBird](#).

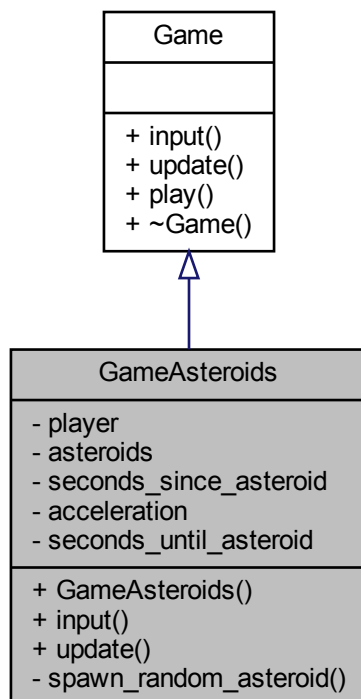
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [game.h](#)
- [game.cpp](#)

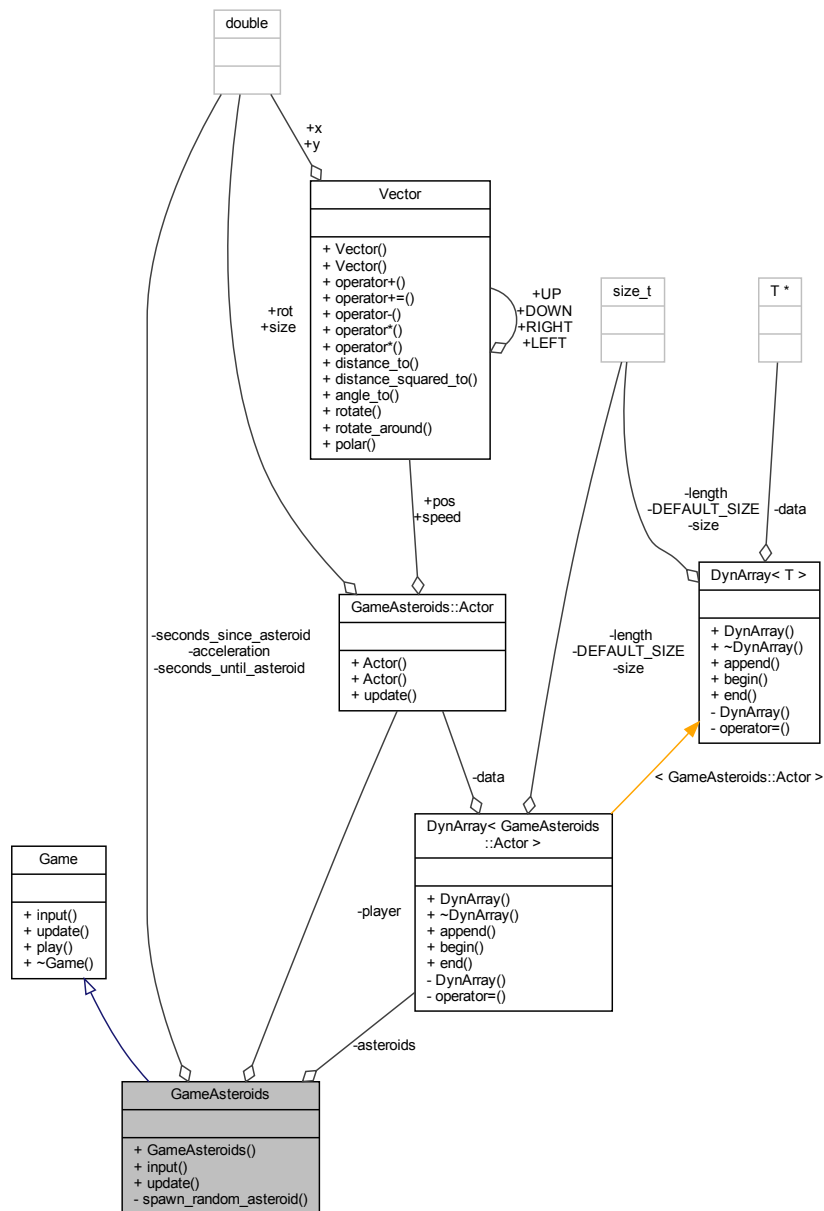
6.7. GameAsteroids osztályreferencia

```
#include <game_asteroids.h>
```

A GameAsteroids osztály származási diagramja:



A GameAsteroids osztály együttműködési diagramja:



Osztályok

- struct [Actor](#)

Publikus tagfüggvények

- [GameAsteroids](#) ()
- void [input](#) (int code)
- bool [update](#) (double delta, [Screen](#) &screen)

Privát tagfüggvények

- void `spawn_random_asteroid` ()
mindig a pálya szélén idéz

Privát attribútumok

- Actor `player`
- DynArray< Actor > `asteroids`
- double `seconds_since_asteroid`

Statikus privát attribútumok

- constexpr static double `acceleration` = 4
- constexpr static double `seconds_until_asteroid` = 4.0

6.7.1. Részletes leírás

Aszteroidák játék minimális változata.

A játékosnak adott ideig kell túlélnie úgy, hogy egyenes vonalú egyenletes mozgással menő, egyre több aszteroidát kell elkerülnie. Lövés nincs, és bizonyos mennyiségű aszteroidával kezdődik a játék. Az új aszteroidák a pálya szélén keletkeznek. Ami kimegy az egyik oldalon, az megjelenik a másikon.

6.7.2. Konstruktorok és destruktorok dokumentációja

6.7.2.1. GameAsteroids()

```
GameAsteroids::GameAsteroids ( )
```

6.7.3. Tagfüggvények dokumentációja

6.7.3.1. input()

```
void GameAsteroids::input (
    int code ) [virtual]
```

Megvalósítja a következőket: [Game](#).

6.7.3.2. spawn_random_asteroid()

```
void GameAsteroids::spawn_random_asteroid ( ) [private]
```

mindig a pálya szélén idéz

6.7.3.3. update()

```
bool GameAsteroids::update (
    double delta,
    Screen & screen ) [virtual]
```

Paraméterek

<i>delta</i>	az eltelt idő, másodpercben
<i>screen</i>	amire rajzolni kell

Visszatérési érték

hogyan vége van-e a játéknak az update miatt

Megvalósítja a következőket: [Game](#).

6.7.4. Adattagok dokumentációja

6.7.4.1. acceleration

```
constexpr static double GameAsteroids::acceleration = 4 [static], [constexpr], [private]
```

6.7.4.2. asteroids

```
DynArray<Actor> GameAsteroids::asteroids [private]
```

6.7.4.3. player

```
Actor GameAsteroids::player [private]
```

6.7.4.4. seconds_since_asteroid

```
double GameAsteroids::seconds_since_asteroid [private]
```

6.7.4.5. seconds_until_asteroid

```
constexpr static double GameAsteroids::seconds_until_asteroid = 4.0 [static], [constexpr],  
[private]
```

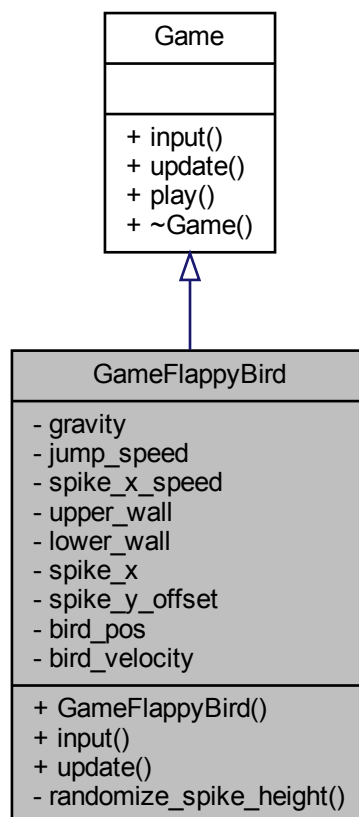
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [game_asteroids.h](#)
- [game_asteroids.cpp](#)

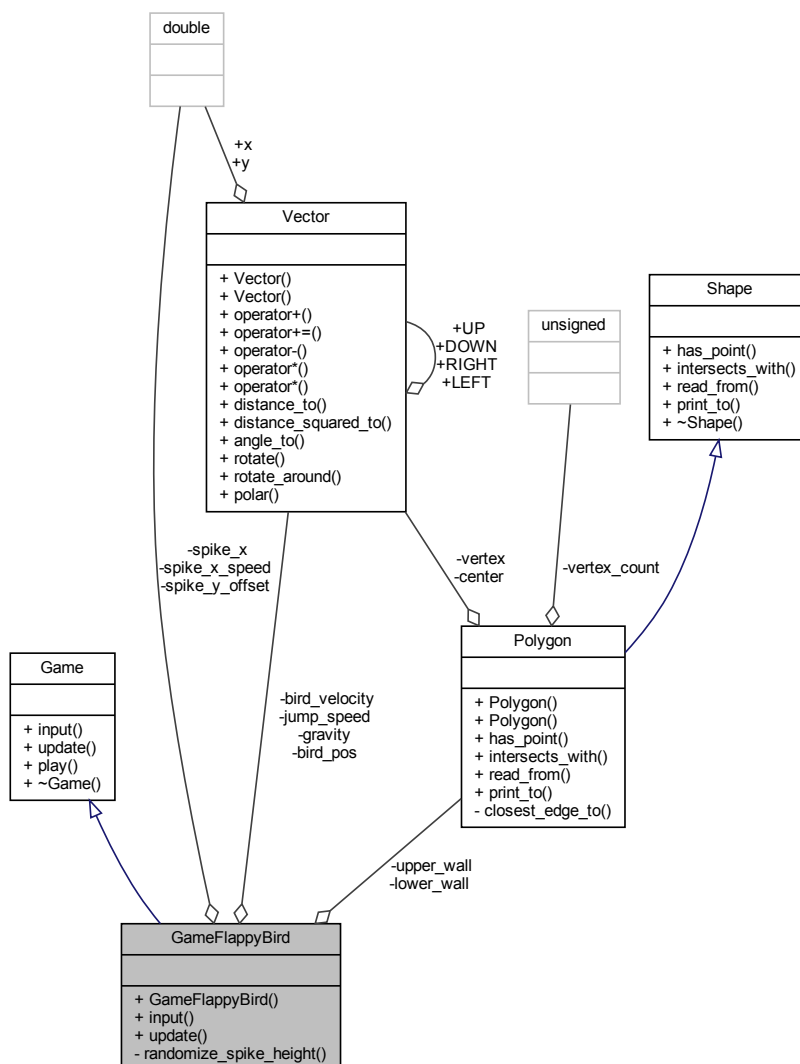
6.8. GameFlappyBird osztályreferencia

```
#include <game_flappy_bird.h>
```

A GameFlappyBird osztály származási diagramja:



A GameFlappyBird osztály együttműködési diagramja:



Publikus tagfüggvények

- [GameFlappyBird](#) ()
- void [input](#) (int code)
- bool [update](#) (double delta, [Screen](#) &screen)

Privát tagfüggvények

- void [randomize_spike_height](#) ()

Privát attribútumok

- const [Vector](#) gravity = [Vector](#)(0, 40)
- const [Vector](#) jump_speed = [Vector](#)(0, -32)
- const double spike_x_speed = -45
- const [Polygon](#) upper_wall = [Polygon](#)([Vector](#)(Screen::size.x / 2, Screen::size.y*0.01 - Screen::size.x / 2), [Vector](#)(0, Screen::size.y*0.01), 4)
- const [Polygon](#) lower_wall = [Polygon](#)([Vector](#)(Screen::size.x / 2, Screen::size.y*0.99 + Screen::size.x / 2), [Vector](#)(0, Screen::size.y*0.99), 4)
- double spike_x
- double spike_y_offset
- [Vector](#) bird_pos
- [Vector](#) bird_velocity

6.8.1. Részletes leírás

Minimális Flappy Bird implementáció.

A játékos nem érhet hozzá a plafonhoz, sem a padlóhoz, sem a tüskékhez. Egyszerre mindig két tüske van a képernyőn, egy fent, egy lent, ezek egyenletesen mozognak jobbról balra. A lyuk magassága valamennyire véletlenszerű, de a mérete állandó.

6.8.2. Konstruktorkok és destruktorkok dokumentációja

6.8.2.1. GameFlappyBird()

```
GameFlappyBird::GameFlappyBird ( )
```

6.8.3. Tagfüggvények dokumentációja

6.8.3.1. input()

```
void GameFlappyBird::input (
    int code ) [virtual]
```

Megvalósítja a következőket: [Game](#).

6.8.3.2. randomize_spike_height()

```
void GameFlappyBird::randomize_spike_height ( ) [private]
```

6.8.3.3. update()

```
bool GameFlappyBird::update (
    double delta,
    Screen & screen ) [virtual]
```

Paraméterek

<i>delta</i>	az eltelt idő, másodpercben
<i>screen</i>	amire rajzolni kell

Visszatérési érték

hogyan vége van-e a játéknak az update miatt

Megvalósítja a következőket: [Game](#).

6.8.4. Adattagok dokumentációja

6.8.4.1. bird_pos

```
Vector GameFlappyBird::bird_pos [private]
```

6.8.4.2. bird_velocity

```
Vector GameFlappyBird::bird_velocity [private]
```

6.8.4.3. gravity

```
const Vector GameFlappyBird::gravity = Vector(0, 40) [private]
```

6.8.4.4. jump_speed

```
const Vector GameFlappyBird::jump_speed = Vector(0, -32) [private]
```

6.8.4.5. lower_wall

```
const Polygon GameFlappyBird::lower_wall = Polygon(Vector(Screen::size.x / 2, Screen::size.y*0.99 + Screen::size.x / 2), Vector(0, Screen::size.y*0.99), 4) [private]
```

6.8.4.6. spike_x

```
double GameFlappyBird::spike_x [private]
```

6.8.4.7. spike_x_speed

```
const double GameFlappyBird::spike_x_speed = -45 [private]
```

6.8.4.8. spike_y_offset

```
double GameFlappyBird::spike_y_offset [private]
```

6.8.4.9. upper_wall

```
const Polygon GameFlappyBird::upper_wall = Polygon(Vector(Screen::size.x / 2, Screen::size.y*0.01 - Screen::size.x / 2), Vector(0, Screen::size.y*0.01), 4) [private]
```

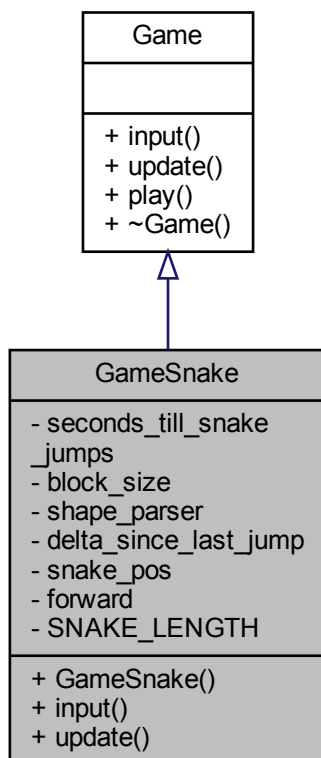
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [game_flappy_bird.h](#)
- [game_flappy_bird.cpp](#)

6.9. GameSnake osztályreferencia

```
#include <game_snake.h>
```

A GameSnake osztály származási diagramja:



6.9.1. Részletes leírás

Minimális Snake játék implementáció.

A játékos fix (10) hosszú, és ez nem változik a játék időtartama alatt. A cél túlélni a kitűzött időtartamig (20s alapértelmezetten). Lebegőpontos számolás és kirajzolás miatt alacsony felbontásnál zavaró lehet, hogy a négyzetek oldalmérete között lehet egy-egy pixel eltérés. A pályát úgy csináltam, hogy elég sok hely legyen az akadályok (egy kör, egy háromszög és egy hatszög), és a 4 fal között, ahol úgy látszik, hogy a kígyó el fog férni, ott tényleg el is fog. A kígyó fejét jelző háromszög mindig az aktuális menetirányba mutat, de lépni csak bizonyos időközönként lép. Emiatt ha a kígyó 180°-ot akarna fordulni, még a következő ugrásig „vissza lehet vonni” a lépést.

6.9.2. Konstruktork és destruktorok dokumentációja

6.9.2.1. GameSnake()

```
GameSnake::GameSnake (
    std::istream & is )
```

6.9.3. Tagfüggvények dokumentációja

6.9.3.1. input()

```
void GameSnake::input (
    int code ) [virtual]
```

Megvalósítja a következőket: [Game](#).

6.9.3.2. update()

```
bool GameSnake::update (
    double delta,
    Screen & screen ) [virtual]
```

Paraméterek

<i>delta</i>	az eltelt idő, másodpercben
<i>screen</i>	amire rajzolni kell

Visszatérési érték

hogy vége van-e a játéknak az update miatt

Megvalósítja a következőket: [Game](#).

6.9.4. Adattagok dokumentációja

6.9.4.1. block_size

```
const double GameSnake::block_size = 4.15 [private]
```

6.9.4.2. delta_since_last_jump

```
double GameSnake::delta_since_last_jump [private]
```

6.9.4.3. forward

```
Vector GameSnake::forward [private]
```

6.9.4.4. seconds_till_snake_jumps

```
const double GameSnake::seconds_till_snake_jumps = 0.2 [private]
```

6.9.4.5. shape_parser

```
ShapeParser GameSnake::shape_parser [private]
```

6.9.4.6. SNAKE_LENGTH

```
const int GameSnake::SNAKE_LENGTH = 10 [static], [private]
```


6.9.4.7. snake_pos

```
Vector GameSnake::snake_pos[SNAKE_LENGTH] [private]
```

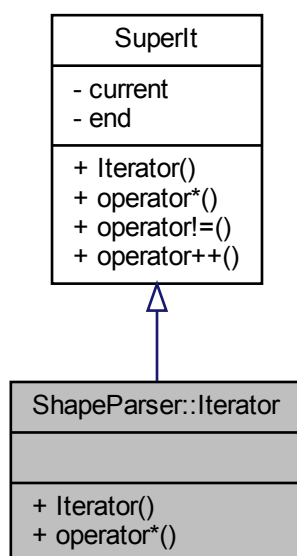
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [game_snake.h](#)
- [game_snake.cpp](#)

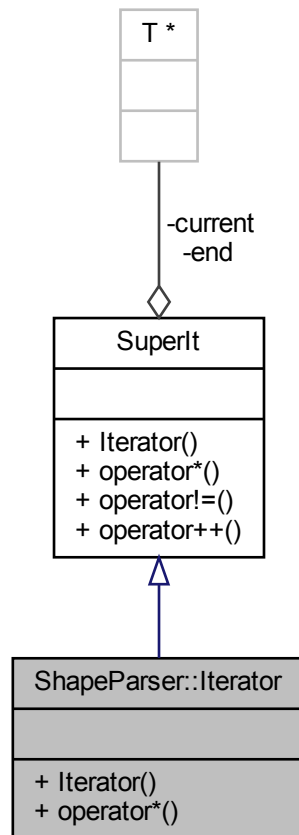
6.10. ShapeParser::Iterator osztályreferencia

```
#include <shape_parser.h>
```

A ShapeParser::Iterator osztály származási diagramja:



A ShapeParser::Iterator osztály együttműködési diagramja:



Publikus tagfüggvények

- `Iterator` (const `SuperIt` &it)
- const `Shape` & `operator*` ()

Privát típusok

- using `value_type` = `std::nullptr_t`
ennek nincs is value_type-ja, Shape absztrakt, megpróbálom így felülrni, hátha így nem engedi az std::algorithmusokat használni rajta

További örökölt tagok

6.10.1. Részletes leírás

Azért kell külön iterátor, mert `Shape`-eken akarunk iterálni, és nem `Shape*`-eken. NEM JÓK AZ ÖRÖKÖLT TYPEDEFJEI! valószínűleg furcsán fog viselkedni `std::algorithmusokkal`, megpróbáltam letiltani. Mivel ennek nincs is value type-ja, mert a `Shape` absztrakt.

6.10.2. Típusdefiníció-tagok dokumentációja

6.10.2.1. value_type

```
using ShapeParser::Iterator::value_type = std::nullptr_t [private]
```

ennek nincs is value_type-ja, [Shape](#) absztrakt, megpróbálom így felülírni, hátha így nem engedi az `std::algorithm`-usokat használni rajta

6.10.3. Konstruktorkok és destruktorkok dokumentációja

6.10.3.1. Iterator()

```
ShapeParser::Iterator::Iterator (
    const SuperIt & it ) [inline]
```

6.10.4. Tagfüggvények dokumentációja

6.10.4.1. operator*()

```
const Shape& ShapeParser::Iterator::operator* ( ) [inline]
```

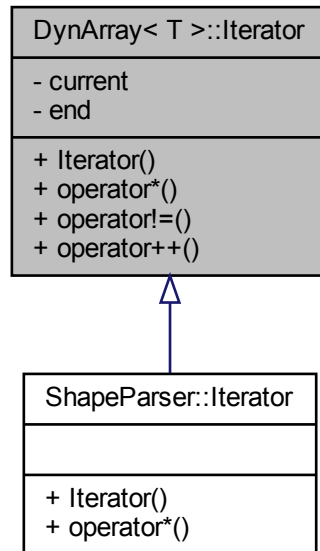
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [shape_parser.h](#)

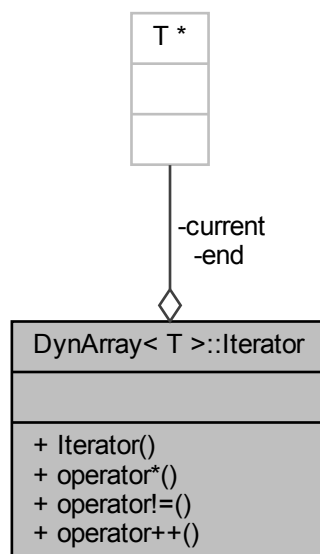
6.11. DynArray< T >::Iterator osztályreferencia

```
#include <dynarray.hpp>
```

A DynArray< T >::Iterator osztály származási diagramja:



A DynArray< T >::Iterator osztály együttműködési diagramja:



Publikus típusok

- using `difference_type` = `std::ptrdiff_t`
azért, hogy std::algoritmusokkal lehessen használni (pl std::equal)
- using `value_type` = `T`
- using `pointer` = `T *`
- using `reference` = `T &`
- using `iterator_category` = `std::output_iterator_tag`

Publikus tagfüggvények

- `Iterator` (`T *current`, `T *end`)
- `T & operator*` ()
- `bool operator!=` (`const Iterator &other`)
- `Iterator & operator++` ()

Privát attribútumok

- `T * current`
- `T * end`

6.11.1. Részletes leírás

```
template<typename T>
class DynArray< T >::Iterator
```

Iterátor a sablon dinamikus tömbhöz.

A túlment iterátor dereferálását kivétellel díjazza.

6.11.2. Típusdefiníció-tagok dokumentációja

6.11.2.1. difference_type

```
template<typename T >
using DynArray< T >::Iterator::difference_type = std::ptrdiff_t
```

azért, hogy `std::algoritmusokkal` lehessen használni (pl `std::equal`)

6.11.2.2. iterator_category

```
template<typename T >
using DynArray< T >::Iterator::iterator_category = std::output_iterator_tag
```

6.11.2.3. pointer

```
template<typename T >
using DynArray< T >::Iterator::pointer = T*
```

6.11.2.4. reference

```
template<typename T >
using DynArray< T >::Iterator::reference = T&
```

6.11.2.5. value_type

```
template<typename T >
using DynArray< T >::Iterator::value_type = T
```

6.11.3. Konstruktorkok és destruktorkok dokumentációja

6.11.3.1. Iterator()

```
template<typename T >
DynArray< T >::Iterator::Iterator (
    T * current,
    T * end ) [inline]
```

6.11.4. Tagfüggvények dokumentációja

6.11.4.1. operator"!=()

```
template<typename T >
bool DynArray< T >::Iterator::operator!= (
    const Iterator & other ) [inline]
```

6.11.4.2. operator*()

```
template<typename T >
T& DynArray< T >::Iterator::operator* ( ) [inline]
```

6.11.4.3. operator++()

```
template<typename T >
Iterator& DynArray< T >::Iterator::operator++ ( ) [inline]
```

6.11.5. Adattagok dokumentációja

6.11.5.1. current

```
template<typename T >
T* DynArray< T >::Iterator::current [private]
```

6.11.5.2. end

```
template<typename T >
T* DynArray< T >::Iterator::end [private]
```

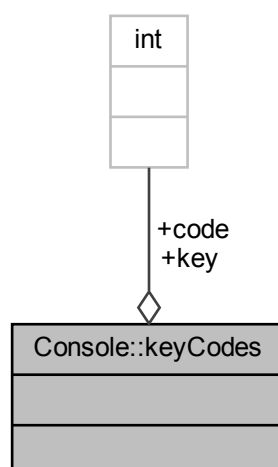
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [dynarray.hpp](#)

6.12. Console::keyCodes struktúrareferencia

Segédtypus a kódváltáshoz.

A Console::keyCodes osztály együttműködési diagramja:



Publikus attribútumok

- int [code](#)
- int [key](#)

6.12.1. Részletes leírás

Segédtypus a kódváltáshoz.

6.12.2. Adattagok dokumentációja

6.12.2.1. code

```
int Console::keyCodes::code
```

6.12.2.2. key

```
int Console::keyCodes::key
```

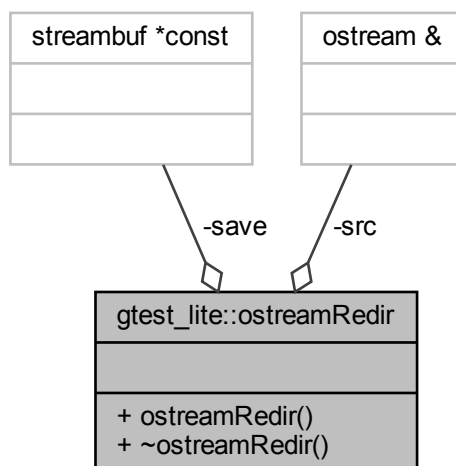
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- [console.cpp](#)

6.13. gtest_lite::ostreamRedir osztályreferencia

```
#include <gtest_lite.h>
```

A gtest_lite::ostreamRedir osztály együttműködési diagramja:



Publikus tagfüggvények

- `ostreamRedir` (`std::ostream &src`, `std::ostream &dst`)
- `~ostreamRedir` ()

Privát attribútumok

- `std::ostream & src`
- `std::streambuf *const save`

6.13.1. Részletes leírás

Segédsablon ostream átirányításához A destruktorkor visszaállít

6.13.2. Konstruktorkok és destruktorkok dokumentációja

6.13.2.1. ostreamRedir()

```
gtest_lite::ostreamRedir::ostreamRedir (
    std::ostream & src,
    std::ostream & dst ) [inline]
```

6.13.2.2. ~ostreamRedir()

```
gtest_lite::ostreamRedir::~~ostreamRedir ( ) [inline]
```

6.13.3. Adattagok dokumentációja

6.13.3.1. save

```
std::streambuf* const gtest_lite::ostreamRedir::save [private]
```

6.13.3.2. src

```
std::ostream& gtest_lite::ostreamRedir::src [private]
```

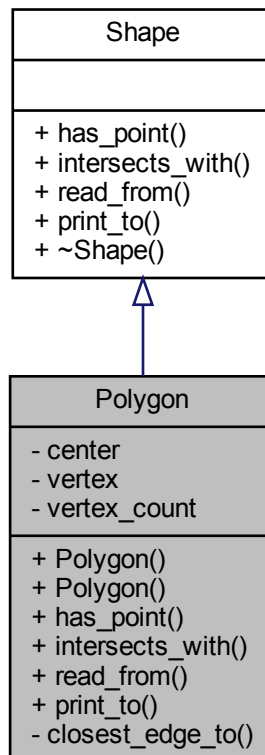
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [gtest_lite.h](#)

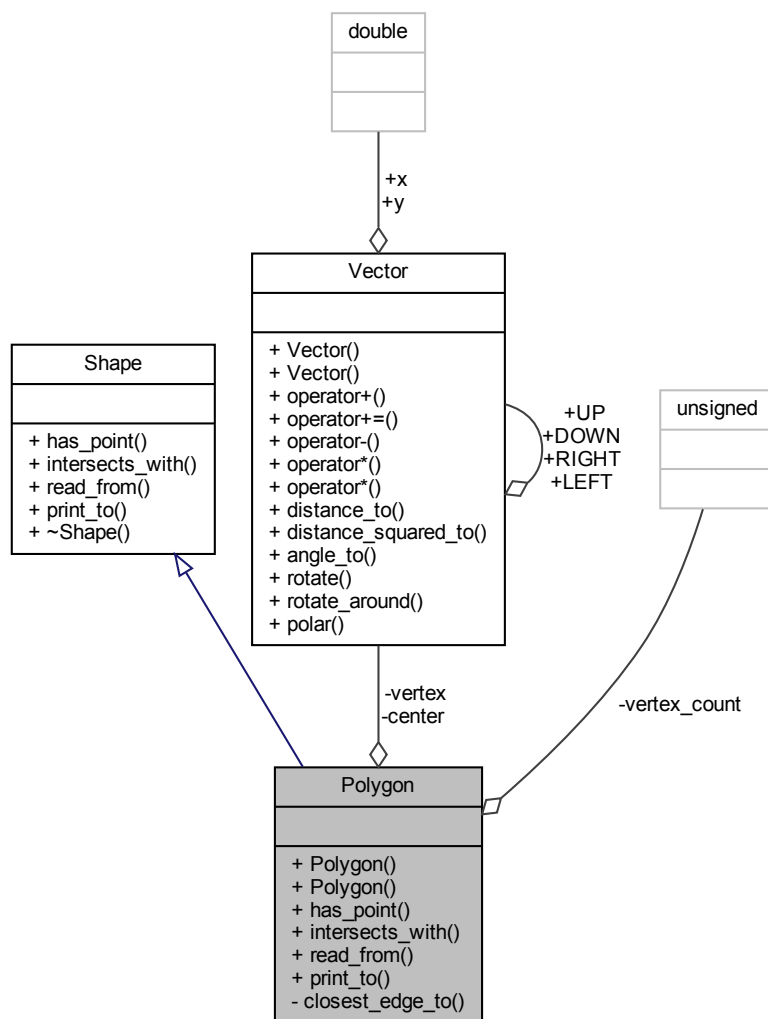
6.14. Polygon osztályreferencia

```
#include <shape_polygon.h>
```

A Polygon osztály származási diagramja:



A Polygon osztály együttműködési diagramja:



Publikus tagfüggvények

- **Polygon** (unsigned vc)
- **Polygon** (Vector center, Vector vertex, unsigned vc)
- bool **has_point** (Vector p) const

Visszaadja, hogy a síkidomban benne van-e p pont.

- bool **intersects_with** (const Circle &c) const

Visszaadja, hogy a síkidomnak és c-nek van-e közös pontja (= metszik-e egymást).

- void **read_from** (std::istream &is)

Beolvassa a síkidomot a bemeneti folyamról, 4 double (center x,y és vertex x,y) formájában.

- void **print_to** (std::ostream &os) const

Kiírja a síkidomot a kimeneti folyamra, emberileg olvasható formában, tehát ez nem alkalmazható a `read_from`-mal közvetlenül perzisztens viselkedés megvalósítására. (A feladat nem is kért ilyet.)

Privát tagfüggvények

- `Segment closest_edge_to (Vector p) const`

Privát attribútumok

- `Vector center`
- `Vector vertex`
- `unsigned vertex_count`

6.14.1. Részletes leírás

A szabályos sokszöget megvalósító osztály.

6.14.2. Konstruktorkok és destruktorok dokumentációja

6.14.2.1. Polygon() [1/2]

```
Polygon::Polygon (
    unsigned vc ) [inline]
```

Majdnem default konstruktor csak az adatfolyamból beolvasás céljára, amúgy memóriaszeméttel inicializál.

Paraméterek

vc	vertex count, négyszög esetén 4 pl, >=3-má lesz téve
----	--

6.14.2.2. Polygon() [2/2]

```
Polygon::Polygon (
    Vector center,
    Vector vertex,
    unsigned vc ) [inline]
```

Paraméterek

vc	vertex count, négyszög esetén 4 pl, >=3-má lesz téve
----	--

6.14.3. Tagfüggvények dokumentációja

6.14.3.1. closest_edge_to()

```
Segment Polygon::closest_edge_to (
    Vector p ) const [private]
```

Visszaadja, hogy a sokszög melyik oldala van az adott ponthoz legközelebb. A visszaadott szakasz b pontja a sokszög középpontjából nézve szög szempontjából pozitívabb irányban van, mint a szakasz a pontja, tehát a szakasz bal oldalán van a sokszög, amennyiben a +y felfelé van.

6.14.3.2. has_point()

```
bool Polygon::has_point (
    Vector p ) const [virtual]
```

Visszaadja, hogy a síkidomban benne van-e p pont.

Megvalósítja a következőket: [Shape](#).

6.14.3.3. intersects_with()

```
bool Polygon::intersects_with (
    const Circle & c ) const [virtual]
```

Visszaadja, hogy a síkidomnak és c-nek van-e közös pontja (= metszik-e egymást).

Megvalósítja a következőket: [Shape](#).

6.14.3.4. print_to()

```
void Polygon::print_to (
    std::ostream & os ) const [inline], [virtual]
```

Kiírja a síkidomot a kimeneti folyamra, emberileg olvasható formában, tehát ez nem alkalmazható a read_from-mal közvetlenül perzisztens viselkedés megvalósítására. (A feladat nem is kért ilyet.)

Megvalósítja a következőket: [Shape](#).

6.14.3.5. read_from()

```
void Polygon::read_from (
    std::istream & is ) [inline], [virtual]
```

Beolvassa a síkidomot a bemeneti folyamról, 4 double (center x,y és vertex x,y) formájában.

Megvalósítja a következőket: [Shape](#).

6.14.4. Adattagok dokumentációja

6.14.4.1. center

```
Vector Polygon::center [private]
```

6.14.4.2. vertex

```
Vector Polygon::vertex [private]
```

6.14.4.3. vertex_count

```
unsigned Polygon::vertex_count [private]
```

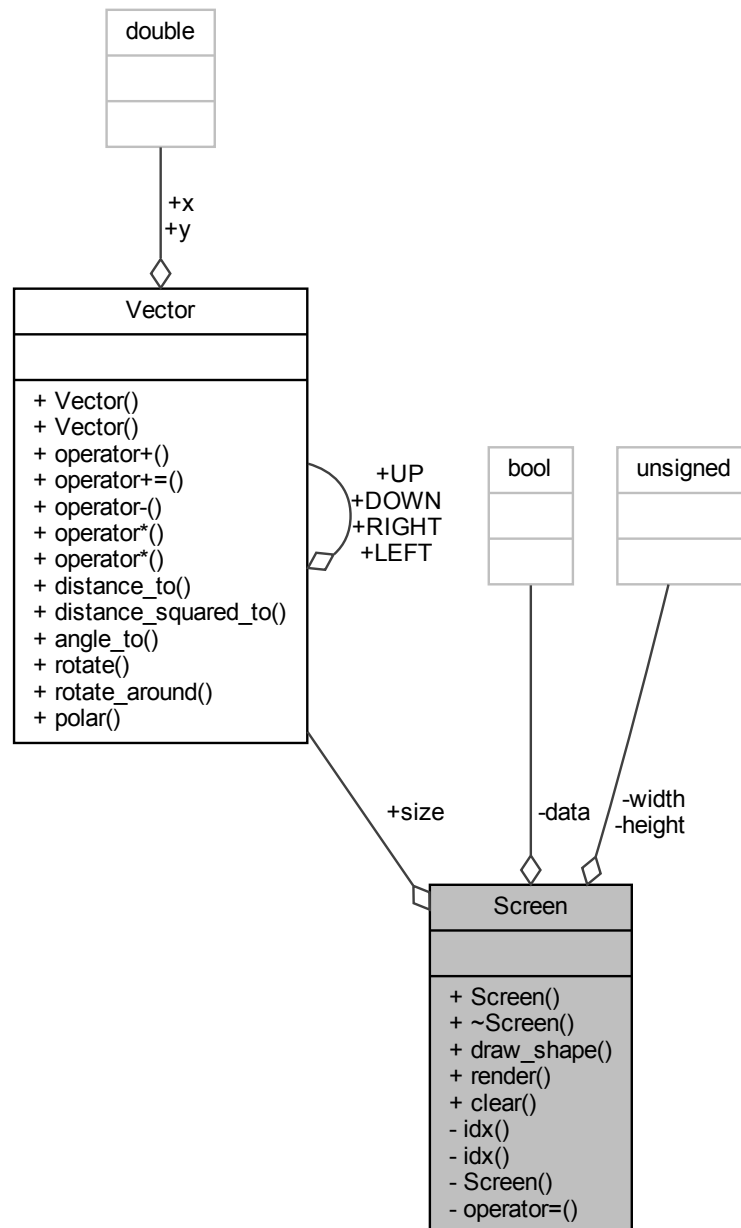
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [shape_polygon.h](#)
- [shape_polygon.cpp](#)

6.15. Screen osztályreferencia

```
#include <screen.h>
```

A Screen osztály együttműködési diagramja:



Publikus tagfüggvények

- [Screen](#) (unsigned [width](#), unsigned [height](#))
- [~Screen](#) ()
- void [draw_shape](#) (const [Shape](#) &s)
- void [render](#) (std::ostream &out) const
Kírja a buffert az adatfolyamra.
- void [clear](#) ()
Törli a buffer minden pixelét.

Statikus publikus attribútumok

- static const `Vector size` = `Vector(80.0, 50.0)`

Privát tagfüggvények

- bool & `idx` (unsigned x, unsigned y)
- bool `idx` (unsigned x, unsigned y) const
- `Screen` (const `Screen` &)
- we don't allow these*
- `Screen` & `operator=` (const `Screen` &)

Privát attribútumok

- const unsigned `width`
- const unsigned `height`
- bool * `data`

6.15.1. Részletes leírás

Az alakzatok lerendereléséért felelős osztály. Egy pixel 2 értéket vehet fel, tehát fekete-fehér a kép.

6.15.2. Konstruktorkok és destruktorkok dokumentációja

6.15.2.1. `Screen()` [1/2]

```
Screen::Screen (
    const Screen & ) [private]
```

we don't allow these

6.15.2.2. `Screen()` [2/2]

```
Screen::Screen (
    unsigned width,
    unsigned height )
```

A paraméterek a terminál felbontására vonatkoznak, tehát például egy 100 oszlopú, 20 sorú konzolablakra `width=100`, `height=40` -es `Screen`-t kell készíteni, ha azt akarjuk, hogy az egészet éppen befedje A magasságot amiatt kell duplázni, mert egy karakter (doboz) 2 pixelnek felel meg (szóköz, alsó doboz, felső doboz vagy teljes doboz) A magasság 2-vel oszthatóvá lesz téve emiatt

6.15.2.3. ~Screen()

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

6.15.3. Tagfüggvények dokumentációja

6.15.3.1. clear()

```
void Screen::clear ( )
```

Törli a buffer minden pixelét.

6.15.3.2. draw_shape()

```
void Screen::draw_shape (
    const Shape & s )
```

A bufferbe belerajzolja az alakzatot, figyelembe véve a virtuális és az igazi felbontás közötti összefüggést. Lehet egymás után több alakzatot is rajzolni, ezek átfedésénél a határt közöttük utólag nem lehet megkülönböztetni.

6.15.3.3. idx() [1/2]

```
bool & Screen::idx (
    unsigned x,
    unsigned y ) [private]
```

6.15.3.4. idx() [2/2]

```
bool Screen::idx (
    unsigned x,
    unsigned y ) const [private]
```

6.15.3.5. operator=()

```
Screen& Screen::operator= (
    const Screen & ) [private]
```

6.15.3.6. render()

```
void Screen::render (
    std::ostream & out ) const
```

Kiírja a buffert az adatfolyamra.

6.15.4. Adattagok dokumentációja

6.15.4.1. data

```
bool* Screen::data [private]
```

6.15.4.2. height

```
const unsigned Screen::height [private]
```

6.15.4.3. size

```
const Vector Screen::size = Vector(80.0, 50.0) [static]
```

A képernyő rajzoláskor ekkorának néz ki, azért, hogy felbontásfüggetlen legyen. Tehát például a (40,25) középpontú kör mindig a képernyő közepén lesz, függetlenül a konzol felbontásától.

6.15.4.4. width

```
const unsigned Screen::width [private]
```

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

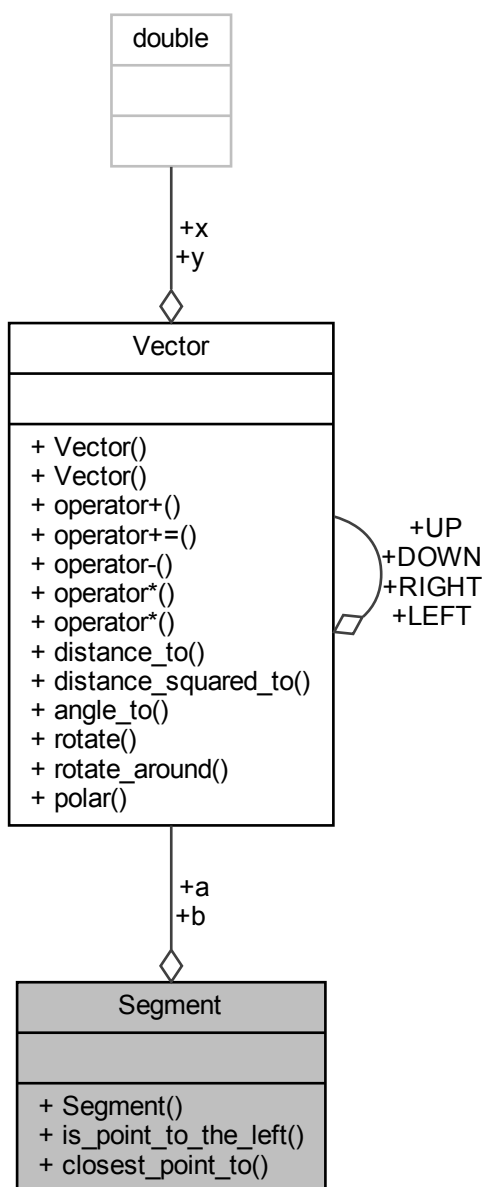
- [screen.h](#)
- [screen.cpp](#)

6.16. Segment struktúráreferencia

egy szakaszt, vagy egyenest reprezentál

```
#include <segment.h>
```

A Segment osztály együttműködési diagramja:



Publikus tagfüggvények

- [Segment](#) ([Vector a](#), [Vector b](#))
- `bool is_point_to_the_left` ([Vector p](#)) `const`
- [Vector closest_point_to](#) ([Vector p](#)) `const`

Publikus attribútumok

- [Vector a](#)
- [Vector b](#)

6.16.1. Részletes leírás

egy szakaszt, vagy egyenest reprezentál

6.16.2. Konstruktork és destruktorok dokumentációja

6.16.2.1. Segment()

```
Segment::Segment (
    Vector a,
    Vector b ) [inline]
```

6.16.3. Tagfüggvények dokumentációja

6.16.3.1. closest_point_to()

```
Vector Segment::closest_point_to (
    Vector p ) const
```

Visszaadja, hogy a szakasz mely pontja van *p*-hez legközelebb. A visszatérési pont mindig a szakaszon van, azaz mindig *a* és *b* között.

6.16.3.2. is_point_to_the_left()

```
bool Segment::is_point_to_the_left (
    Vector p ) const
```

Visszaadja, hogy bal oldalon van-e a pont, amennyiben a pozitív forgásirány óramutató járásával ellenkező (azaz +y felfelé van). Ha óramutatóval megegyező / +y lefelé van (pl képernyő), akkor azt mondja meg, hogy jobbra van-e a pont.

6.16.4. Adattagok dokumentációja

6.16.4.1. a

```
Vector Segment::a
```

6.16.4.2. b

```
Vector Segment::b
```

Ez a dokumentáció a struktúráról a következő fájlok alapján készült:

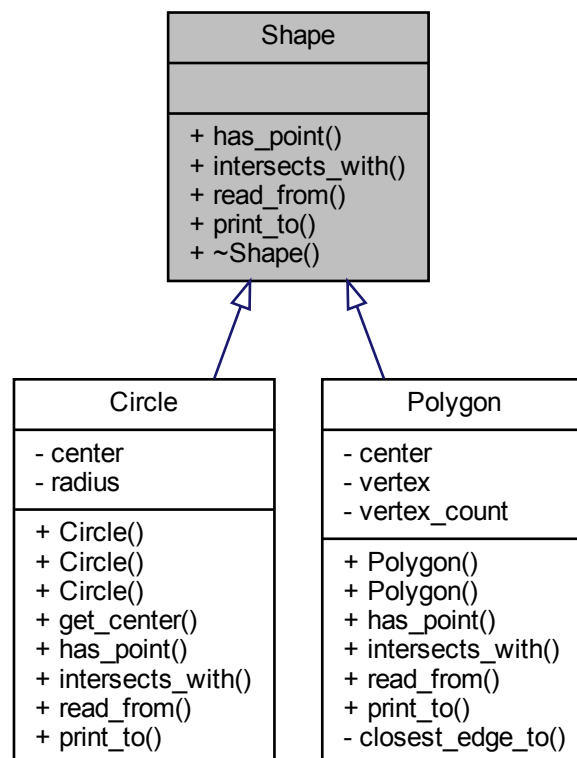
- [segment.h](#)
- [segment.cpp](#)

6.17. Shape osztályreferencia

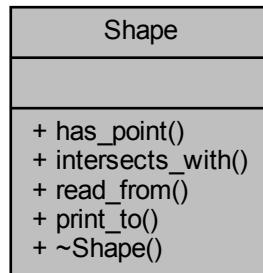
Absztrakt síkidom őssztály.

```
#include <shape.h>
```

A Shape osztály származási diagramja:



A Shape osztály együttműködési diagramja:



Publikus tagfüggvények

- virtual bool `has_point` (Vector p) const =0
Visszaadja, hogy a síkidomban benne van-e p pont.
- virtual bool `intersects_with` (const Circle &c) const =0
Visszaadja, hogy a síkidomnak és c-nek van-e közös pontja (= metszik-e egymást).
- virtual void `read_from` (std::istream &is)=0
Beolvassa a síkidomot a bemeneti folyamról, 4 double (center x,y és vertex x,y) formájában.
- virtual void `print_to` (std::ostream &os) const =0
Kiírja a síkidomot a kimeneti folyamra, emberileg olvasható formában, tehát ez nem alkalmazható a read_from-mal közvetlenül perzisztens viselkedés megvalósítására. (A feladat nem is kért ilyet.)
- virtual `~Shape` ()

6.17.1. Részletes leírás

Absztrakt síkidom ősosztály.

6.17.2. Konstruktorok és destruktorok dokumentációja

6.17.2.1. ~Shape()

```
virtual Shape::~Shape ( ) [inline], [virtual]
```

6.17.3. Tagfüggvények dokumentációja

6.17.3.1. has_point()

```
virtual bool Shape::has_point (
    Vector p ) const [pure virtual]
```

Visszaadja, hogy a síkidomban benne van-e p pont.

Megvalósítják a következők: [Polygon](#) és [Circle](#).

6.17.3.2. intersects_with()

```
virtual bool Shape::intersects_with (
    const Circle & c ) const [pure virtual]
```

Visszaadja, hogy a síkidomnak és c-nek van-e közös pontja (= metszik-e egymást).

Megvalósítják a következők: [Polygon](#) és [Circle](#).

6.17.3.3. print_to()

```
virtual void Shape::print_to (
    std::ostream & os ) const [pure virtual]
```

Kiírja a síkidomot a kimeneti folyamra, emberileg olvasható formában, tehát ez nem alkalmazható a read_from-mal közvetlenül perzisztens viselkedés megvalósítására. (A feladat nem is kért ilyet.)

Megvalósítják a következők: [Circle](#) és [Polygon](#).

6.17.3.4. read_from()

```
virtual void Shape::read_from (
    std::istream & is ) [pure virtual]
```

Beolvassa a síkidomot a bemeneti folyamról, 4 double (center x,y és vertex x,y) formájában.

Megvalósítják a következők: [Circle](#) és [Polygon](#).

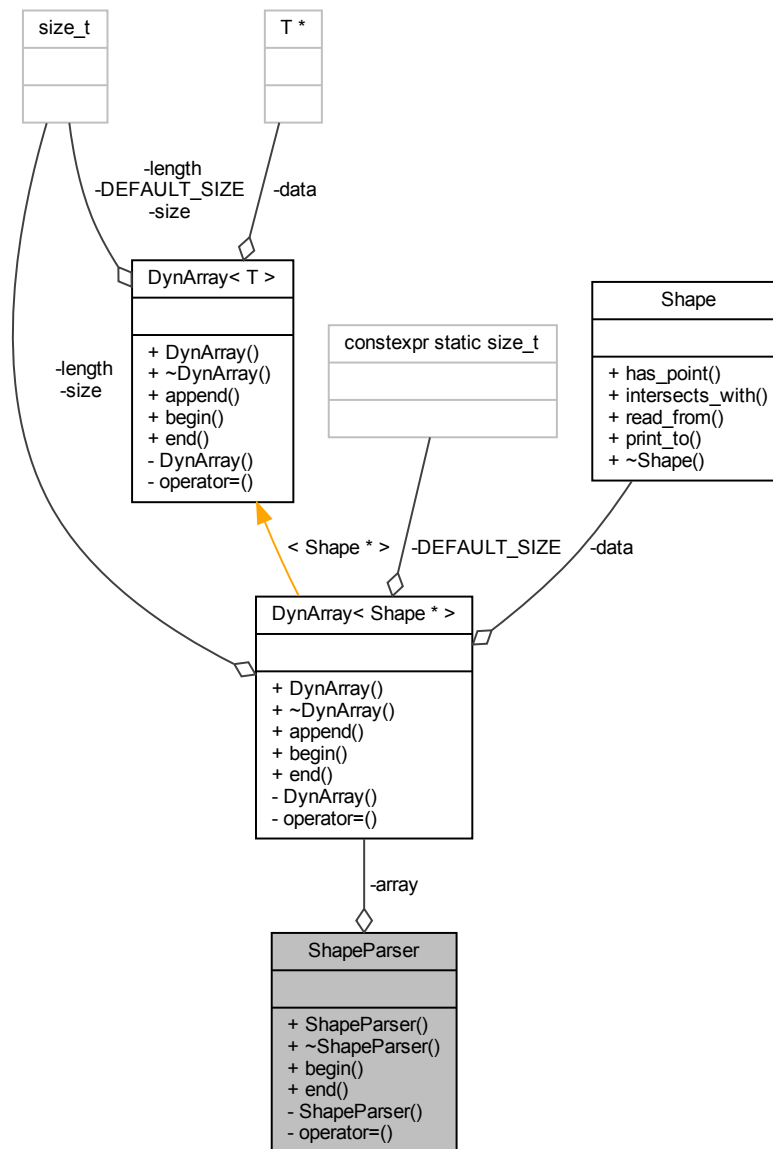
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [shape.h](#)

6.18. ShapeParser osztályreferencia

```
#include <shape_parser.h>
```

A ShapeParser osztály együttműködési diagramja:



Osztályok

- class [Iterator](#)

Publikus tagfüggvények

- [ShapeParser](#) (std::istream &is, bool(*pred)(const [Shape](#) &)=[])(const [Shape](#) &) { return true;}}
- [~ShapeParser](#) ()
- [Iterator](#) begin ()
- [Iterator](#) end ()

Privát típusok

- typedef `DynArray< Shape * >::Iterator` `SuperIt`

Privát tagfüggvények

- `ShapeParser` (const `ShapeParser` &)
- `ShapeParser` & `operator=` (const `ShapeParser` &)

Privát attribútumok

- `DynArray< Shape * >` `array`

6.18.1. Részletes leírás

Az alakzatbeolvasó osztály.

Saját dinamikus memóriáját kezeli, és felhasználja a dinamikus tömböt a pointerok tárolására. Adatok elérése kizárólag iterátorral, a beolvasott adatok nem módosíthatóak.

6.18.2. Típusdefiníció-tagok dokumentációja

6.18.2.1. SuperIt

```
typedef DynArray<Shape*>::Iterator ShapeParser::SuperIt [private]
```

6.18.3. Konstruktorok és destruktorok dokumentációja

6.18.3.1. ShapeParser() [1/2]

```
ShapeParser::ShapeParser (
    const ShapeParser & ) [private]
```

6.18.3.2. ShapeParser() [2/2]

```
ShapeParser::ShapeParser (
    std::istream & is,
    bool(*) (const Shape &) pred = [] (const Shape&) { return true; } )
```

Konstruktor, és egyben beolvasó függvény is. RAI, így a konstruktorban foglalja a dinamikus memóriáját.

Paraméterek

<i>pred</i>	predikátum, meg lehet mondani, hogy eltávoljuk-e az adott alakzatot
<i>is</i>	ahonnét az alakzatokat olvassa, addig, amíg nem lesz hiba. Sikeres beolvasást <code>is.eof()</code> -al lehet (és érdemes) tesztelni.

6.18.3.3. ~ShapeParser()

```
ShapeParser::~~ShapeParser ( )
```

6.18.4. Tagfüggvények dokumentációja**6.18.4.1. begin()**

```
Iterator ShapeParser::begin ( ) [inline]
```

6.18.4.2. end()

```
Iterator ShapeParser::end ( ) [inline]
```

6.18.4.3. operator=()

```
ShapeParser& ShapeParser::operator= (
    const ShapeParser & ) [private]
```

6.18.5. Adattagok dokumentációja**6.18.5.1. array**

```
DynArray<Shape*> ShapeParser::array [private]
```

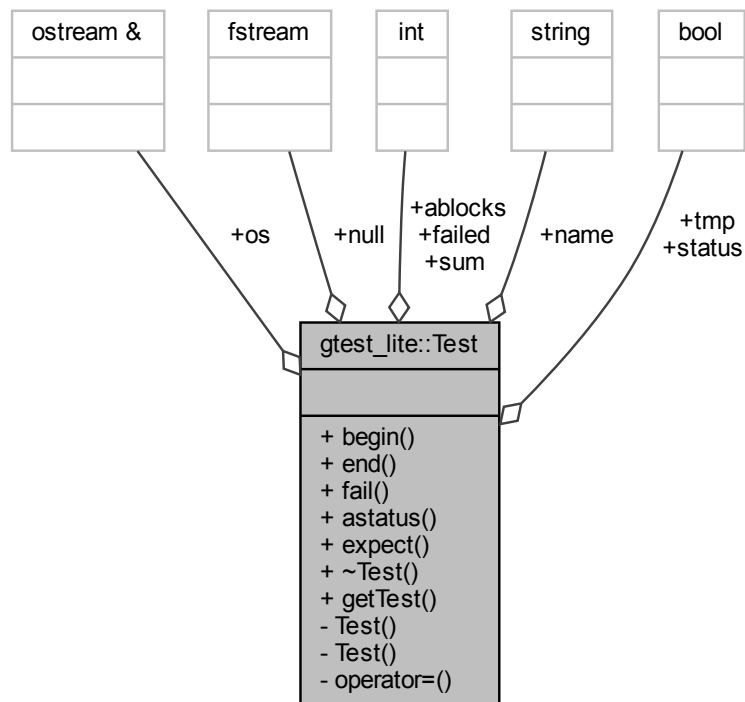
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [shape_parser.h](#)
- [shape_parser.cpp](#)

6.19. gtest_lite::Test struktúráreferencia

```
#include <gtest_lite.h>
```

A gtest_lite::Test osztály együttműködési diagramja:



Publikus tagfüggvények

- void **begin** (const char *n)
Teszt kezdete.
- std::ostream & **end** (bool memchk=false)
Teszt vége.
- bool **fail** ()
- bool **astatus** ()
- std::ostream & **expect** (bool st, const char *file, int line, const char *expr, bool pr=false)
Eredményt adminisztráló tagfüggvény True a jó eset.
- **~Test** ()
Destruktor.

Statikus publikus tagfüggvények

- static **Test** & **getTest** ()

Publikus attribútumok

- int `sum`
tesztek számlálója
- int `failed`
hibás tesztek
- int `ablocks`
allokált blokkok száma
- bool `status`
éppen futó teszt státusza.
- bool `tmp`
temp a kivételkezeléshez;
- std::string `name`
éppen futó teszt neve.
- std::fstream `null`
nyelő, ha nem kell kiírni semmit
- std::ostream & `os`
ide írunk

Privát tagfüggvények

- `Test ()`
singleton minta miatt
- `Test (const Test &)`
- `void operator= (const Test &)`

6.19.1. Részletes leírás

Tesztek állapotát tároló osztály. Egyetlen egy statikus példány keletkezik, aminek a destruktora a futás végén hívódik meg.

6.19.2. Konstruktorkok és destruktorkok dokumentációja

6.19.2.1. `Test()` [1/2]

```
gtest_lite::Test::Test ( ) [inline], [private]
```

singleton minta miatt

6.19.2.2. `Test()` [2/2]

```
gtest_lite::Test::Test (
    const Test & ) [private]
```

6.19.2.3. ~Test()

```
gtest_lite::Test::~~Test ( ) [inline]
```

Destruktor.

6.19.3. Tagfüggvények dokumentációja

6.19.3.1. astatus()

```
bool gtest_lite::Test::astatus ( ) [inline]
```

6.19.3.2. begin()

```
void gtest_lite::Test::begin (
    const char * n ) [inline]
```

Teszt kezdete.

6.19.3.3. end()

```
std::ostream& gtest_lite::Test::end (
    bool memchk = false ) [inline]
```

Teszt vége.

6.19.3.4. expect()

```
std::ostream& gtest_lite::Test::expect (
    bool st,
    const char * file,
    int line,
    const char * expr,
    bool pr = false ) [inline]
```

Eredményt adminisztráló tagfüggvény True a jó eset.

6.19.3.5. fail()

```
bool gtest_lite::Test::fail ( ) [inline]
```

6.19.3.6. getTest()

```
static Test& gtest_lite::Test::getTest ( ) [inline], [static]
```

< egyedüli (singleton) példány

6.19.3.7. operator=()

```
void gtest_lite::Test::operator= (
    const Test & ) [private]
```

6.19.4. Adattagok dokumentációja

6.19.4.1. ablocks

```
int gtest_lite::Test::ablocks
```

allokált blokkok száma

6.19.4.2. failed

```
int gtest_lite::Test::failed
```

hibás tesztek

6.19.4.3. name

```
std::string gtest_lite::Test::name
```

éppen futó teszt neve.

6.19.4.4. null

```
std::fstream gtest_lite::Test::null
```

nyelő, ha nem kell kiírni semmit

6.19.4.5. os

```
std::ostream& gtest_lite::Test::os
```

ide írunk

6.19.4.6. status

```
bool gtest_lite::Test::status
```

éppen futó teszt státusza.

6.19.4.7. sum

```
int gtest_lite::Test::sum
```

tesztek számlálója

6.19.4.8. tmp

```
bool gtest_lite::Test::tmp
```

temp a kivételkezeléshez;

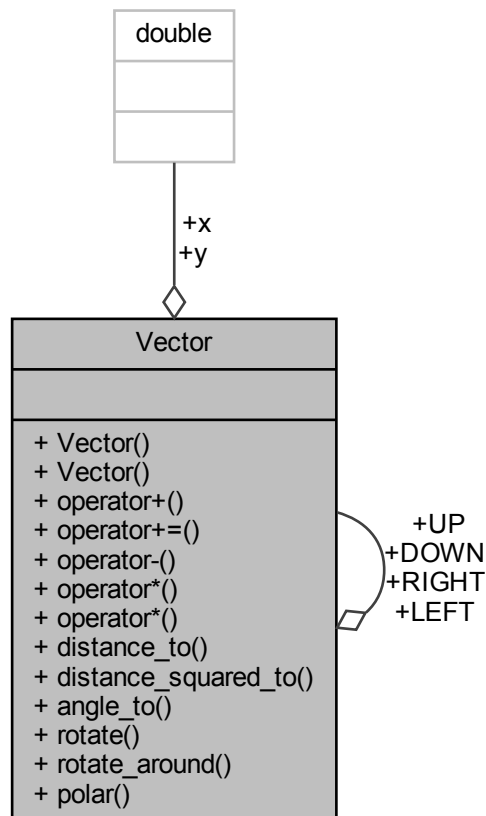
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- [gtest_lite.h](#)

6.20. Vector struktúrareferencia

```
#include <vector.h>
```

A Vector osztály együttműködési diagramja:



Publikus tagfüggvények

- `Vector ()`
Default konstruktor, csak streamből beolvasás céljára. Memóriaszeméttel inicializál.
- `Vector (double x, double y)`
Descartes-koordinátás konstruktor.
- `Vector operator+ (Vector other) const`
- `Vector & operator+= (Vector other)`
- `Vector operator- (Vector other) const`
- `Vector operator* (double other) const`
- `double operator* (Vector other) const`
- `double distance_to (Vector v) const`
- `double distance_squared_to (Vector v) const`
- `double angle_to (Vector v) const`
radiánt ad, jobbra van a 0, a pozitív irány a +x tengelytől a +y tengely irányába mutat
- `void rotate (double angle)`
- `void rotate_around (Vector c, double angle)`

Statikus publikus tagfüggvények

- static `Vector polar` (double r, double angle)
Polárkoordinátás konstruktor-szerűség.

Publikus attribútumok

- double `x`
- double `y`

Statikus publikus attribútumok

- static const `Vector UP`
- static const `Vector DOWN`
- static const `Vector LEFT`
- static const `Vector RIGHT`

6.20.1. Részletes leírás

Két dimenziós, lebegőpontos (double) vektor (= irányított szakasz) megvalósítása.

6.20.2. Konstruktorok és destruktorok dokumentációja

6.20.2.1. `Vector()` [1/2]

```
Vector::Vector ( ) [inline]
```

Default konstruktor, csak streamből beolvasás céljára. Memóriaszeméttel inicializál.

6.20.2.2. `Vector()` [2/2]

```
Vector::Vector (
    double x,
    double y ) [inline]
```

Descartes-koordinátás konstruktor.

6.20.3. Tagfüggvények dokumentációja

6.20.3.1. angle_to()

```
double Vector::angle_to (
    Vector v ) const [inline]
```

radiánt ad, jobbra van a 0, a pozitív irány a +x tengelytől a +y tengely irányába mutat

6.20.3.2. distance_squared_to()

```
double Vector::distance_squared_to (
    Vector v ) const [inline]
```

6.20.3.3. distance_to()

```
double Vector::distance_to (
    Vector v ) const [inline]
```

6.20.3.4. operator*() [1/2]

```
Vector Vector::operator* (
    double other ) const [inline]
```

6.20.3.5. operator*() [2/2]

```
double Vector::operator* (
    Vector other ) const [inline]
```

6.20.3.6. operator+()

```
Vector Vector::operator+ (
    Vector other ) const [inline]
```

6.20.3.7. operator+=()

```
Vector& Vector::operator+= (
    Vector other ) [inline]
```

6.20.3.8. operator-()

```
Vector Vector::operator- (
    Vector other ) const [inline]
```

6.20.3.9. polar()

```
static Vector Vector::polar (
    double r,
    double angle ) [inline], [static]
```

Polárkoordinátás konstruktor-szerűség.

6.20.3.10. rotate()

```
void Vector::rotate (
    double angle ) [inline]
```

Paraméterek

<i>angle</i>	radián, jobbra van a 0, a pozitív irány a +x tengelytől a +y tengely irányába mutat
--------------	---

6.20.3.11. rotate_around()

```
void Vector::rotate_around (
    Vector c,
    double angle ) [inline]
```

Paraméterek

<i>c</i>	a forgatás középpontja
<i>angle</i>	radián, jobbra van a 0, a pozitív irány a +x tengelytől a +y tengely irányába mutat

6.20.4. Adattagok dokumentációja

6.20.4.1. DOWN

```
const Vector Vector::DOWN [static]
```

6.20.4.2. LEFT

```
const Vector Vector::LEFT [static]
```

6.20.4.3. RIGHT

```
const Vector Vector::RIGHT [static]
```

6.20.4.4. UP

```
const Vector Vector::UP [static]
```

6.20.4.5. x

```
double Vector::x
```

6.20.4.6. y

```
double Vector::y
```

Ez a dokumentáció a struktúráról a következő fájlok alapján készült:

- [vector.h](#)
- [vector.cpp](#)

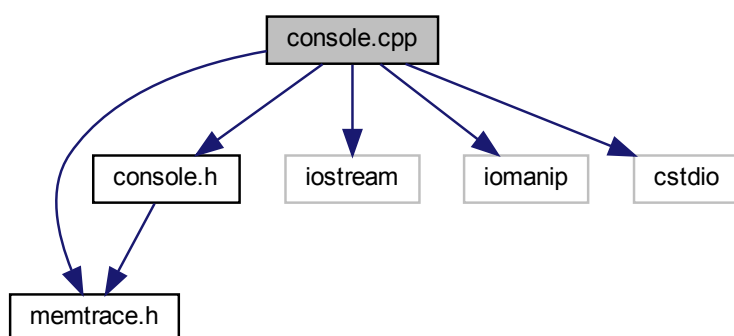
7. fejezet

Fájlok dokumentációja

7.1. console.cpp fájlreferencia

```
#include "memtrace.h"  
#include "console.h"  
#include <iostream>  
#include <iomanip>  
#include <cstdio>
```

A console.cpp definíciós fájl függési gráfja:



Osztályok

- struct [Console::keyCodes](#)
Segéd típus a kódváltáshoz.

Névterek

- [anonymous_namespace{console.cpp}](#)

Makródefiníciók

- `#define C(c, k) {c,k}`

7.1.1. Részletes leírás

[Console](#) input/output kezelése Linux/UNIX alatt csak ANSI terminálbeállításokkal megy. (ncurses/pdcurses használatával hordozható lehetne! Help welcome!)

7.1.2. Makródefiníciók dokumentációja

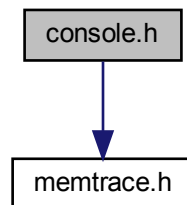
7.1.2.1. C

```
#define C(  
    c,  
    k ) {c,k}
```

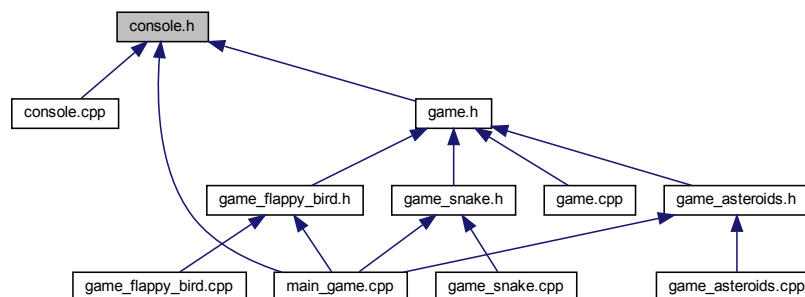
7.2. console.h fájlreferencia

```
#include "memtrace.h"
```

A console.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [Console](#)

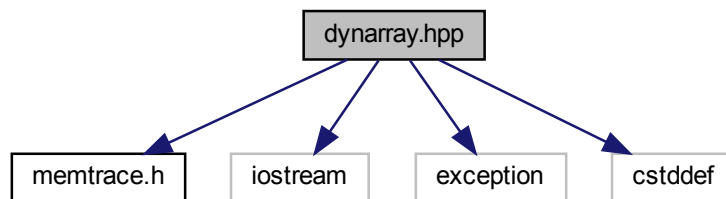
7.2.1. Részletes leírás

[Console](#) input/output kezelése

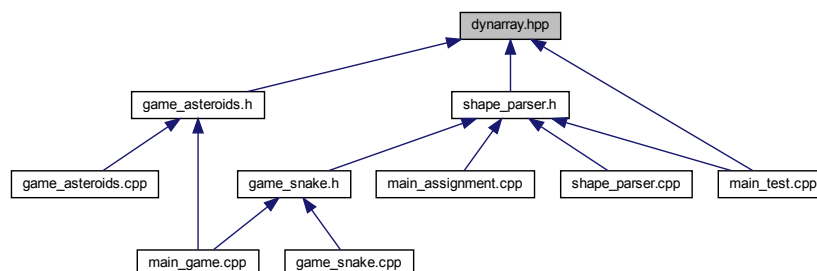
7.3. dynarray.hpp fájlreferencia

```
#include "memtrace.h"
#include <iostream>
#include <exception>
#include <cstdint>
```

A dynarray.hpp definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [DynArray< T >](#)
- class [DynArray< T >::iterator](#)

7.3.1. Részletes leírás

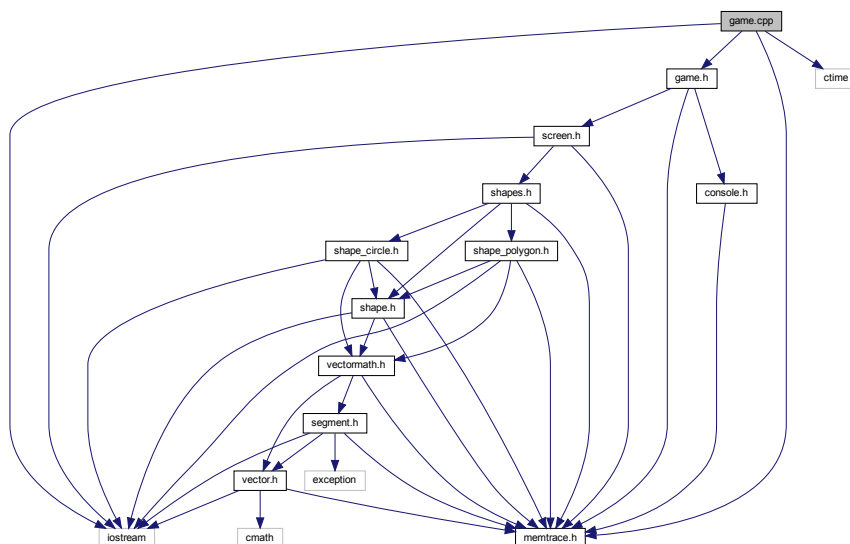
Sablonos dinamikus tömb megvalósítása.

Az elemek csak iterátorral érhetők el, és hozzáfűzni lehet csak, törölni nem. Ha pointer típust tárol dinamikus memóriára, akkor a használó feladata a felszabadítás.

7.4. game.cpp fájlreferencia

```
#include "memtrace.h"
#include <iostream>
#include <ctime>
#include "game.h"
```

A game.cpp definíciós fájl függési gráfja:



Változók

- const double `max_delta` = 0.1

7.4.1. Részletes leírás

Minden játék főciklusa, általánosítva.

7.4.2. Változók dokumentációja

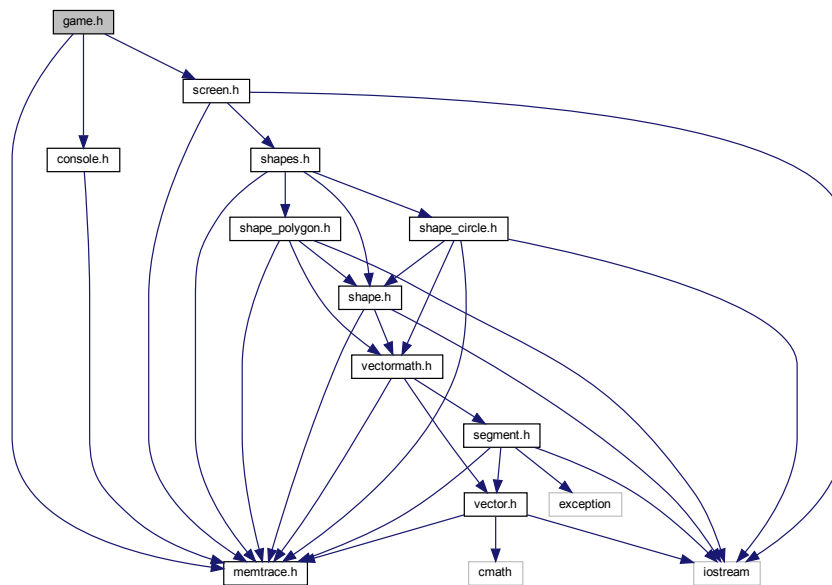
7.4.2.1. max_delta

```
const double max_delta = 0.1
```

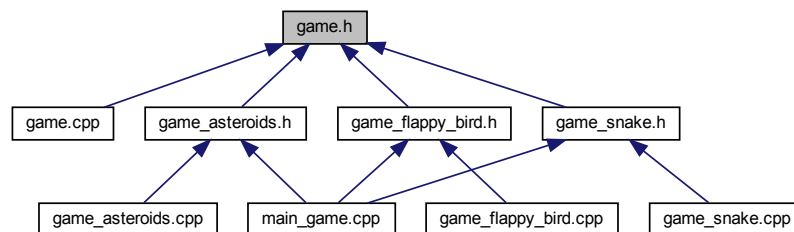
7.5. game.h fájltreferencia

```
#include "memtrace.h"
#include "screen.h"
#include "console.h"
```

A game.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [Game](#)

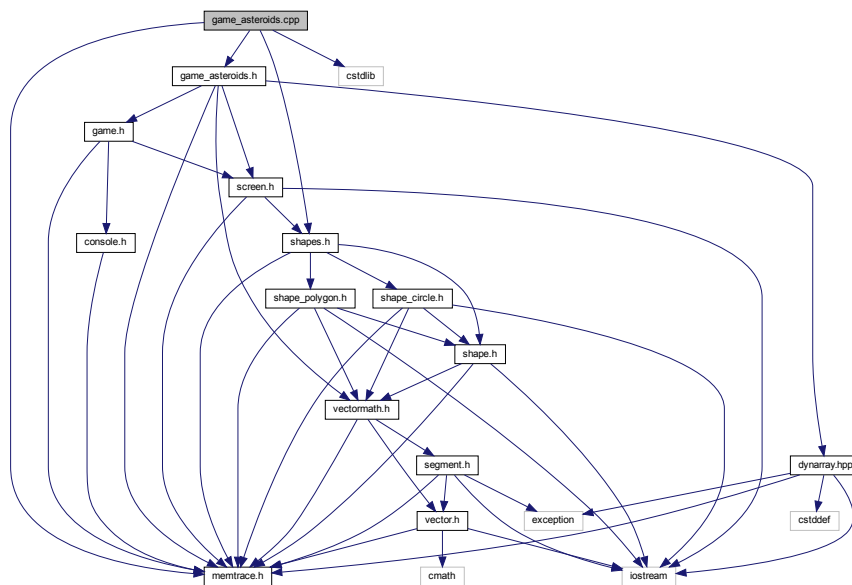
7.5.1. Részletes leírás

Absztrakt játék alaposztály, a közösen kezelhetőség céljából.

7.6. game_asteroids.cpp fájlreferencia

```
#include "memtrace.h"
#include "game_asteroids.h"
#include <cstdlib>
#include "shapes.h"
```

A game_asteroids.cpp definíciós fájl függési gráfja:



Névterek

- [anonymous_namespace{game_asteroids.cpp}](#)

Függvények

- `double anonymous_namespace{game_asteroids.cpp}::randd (double range)`

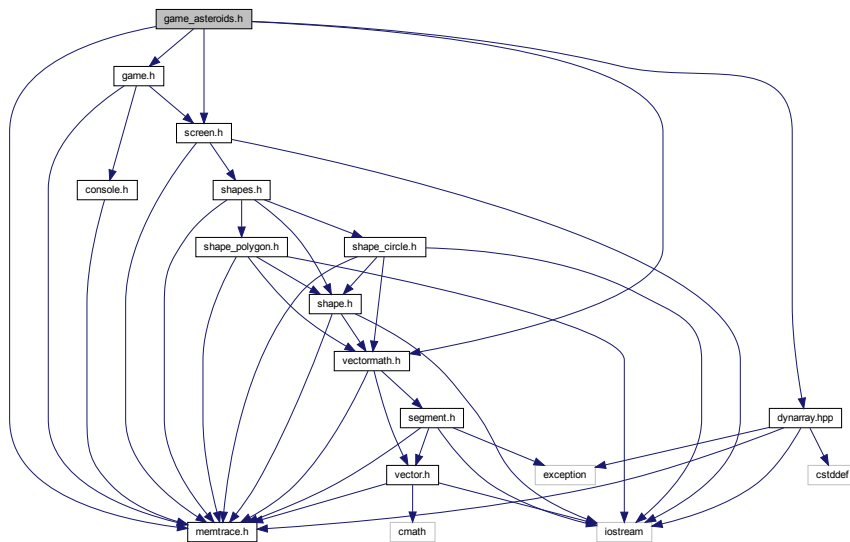
7.6.1. Részletes leírás

Függvények megvalósítása a játékhoz.

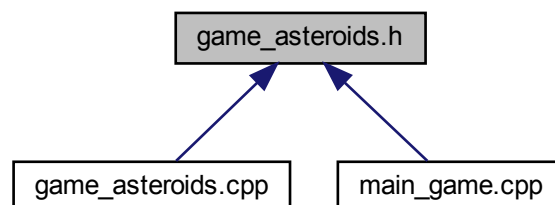
7.7. game_asteroids.h fájlreferencia

```
#include "memtrace.h"
#include "vectormath.h"
#include "screen.h"
#include "dynarray.hpp"
#include "game.h"
```

A game_asteroids.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [GameAsteroids](#)
- struct [GameAsteroids::Actor](#)

7.7.1. Részletes leírás

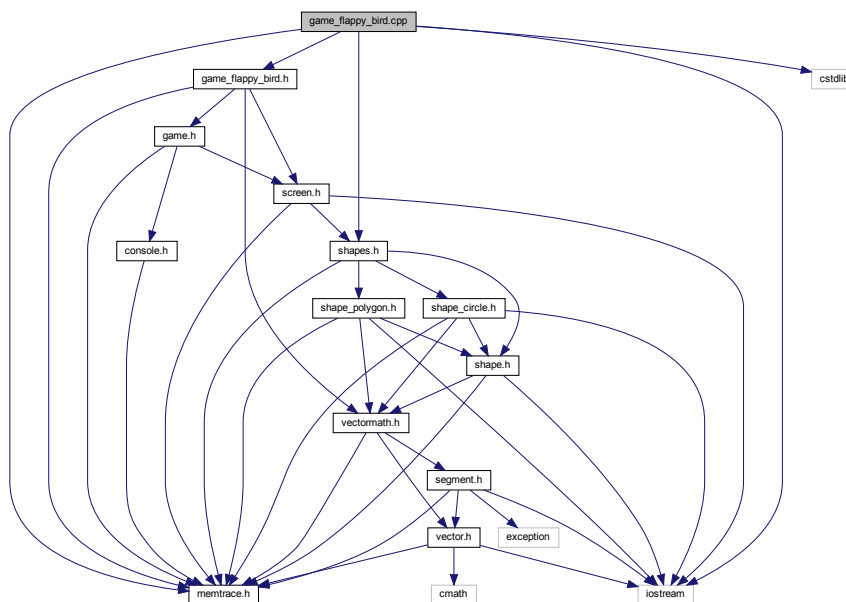
Aszteroidák játék minimális változata.

A játékosnak adott ideig kell túlélnie úgy, hogy egyenes vonalú egyenletes mozgással menő, egyre több aszteroidát kell elkerülnie. Lövés nincs, és bizonyos mennyiségű aszteroidával kezdődik a játék. Az új aszteroidák a pálya szélén keletkeznek. Ami kimegy az egyik oldalon, az megjelenik a másikon.

7.8. game_flappy_bird.cpp fájlreferencia

```
#include "memtrace.h"
#include "game_flappy_bird.h"
#include <iostream>
#include <cstdlib>
#include "shapes.h"
```

A game_flappy_bird.cpp definíciós fájl függési gráfja:



7.8.1. Részletes leírás

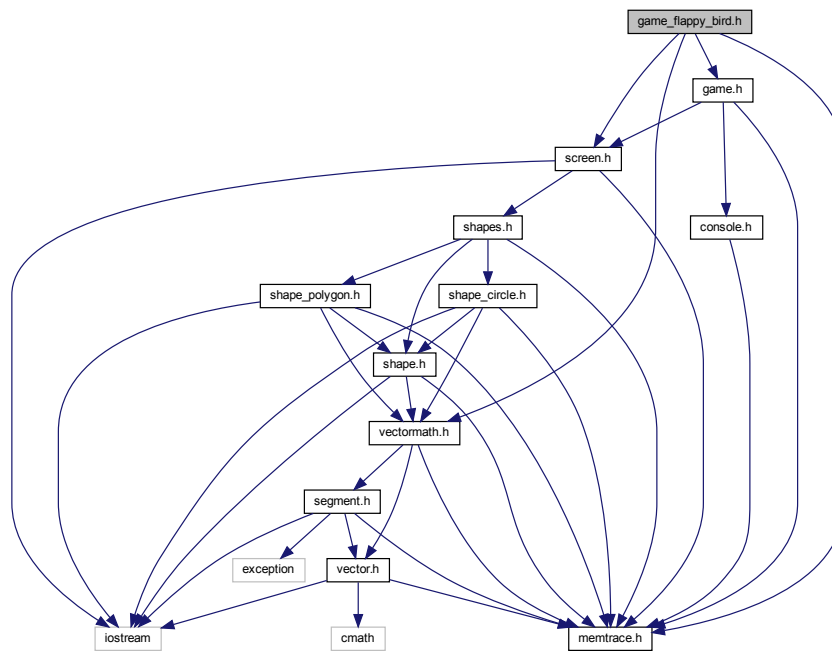
Függvények megvalósítása a játékhoz.

7.9. game_flappy_bird.h fájlreferencia

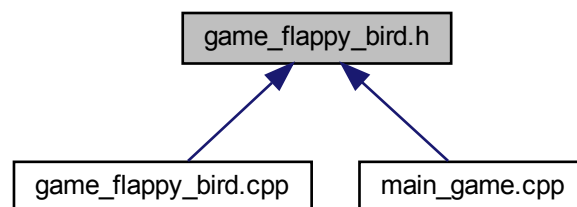
```
#include "memtrace.h"
#include "vectormath.h"
#include "screen.h"
```

```
#include "game.h"
```

A game_flappy_bird.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

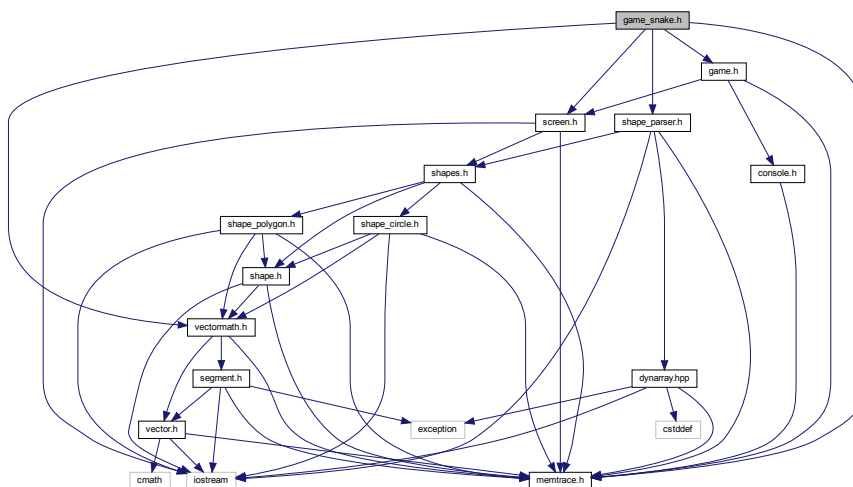
- class [GameFlappyBird](#)

7.9.1. Részletes leírás

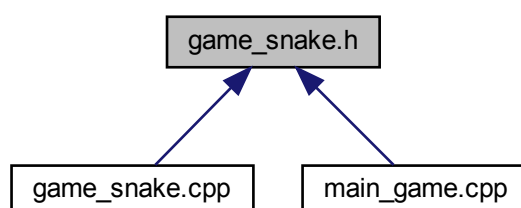
Minimális Flappy Bird implementáció.

A játékos nem érhet hozzá a plafonhoz, sem a padlóhoz, sem a tüskékhez. Egyszerre mindig két tüske van a képernyőn, egy fent, egy lent, ezek egyenletesen mozognak jobbról balra. A lyuk magassága valamennyire véletlenszerű, de a mérete állandó.

A game_snake.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class GameSnake

7.11.1. Részletes leírás

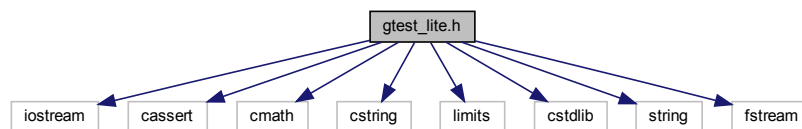
Minimális Snake játék implementáció.

A játékos fix (10) hosszú, és ez nem változik a játék időtartama alatt. A cél túlélni a kitűzött időtartamig (20s alapértelmezetten). Lebegőpontos számolás és kirajzolás miatt alacsony felbontásnál zavaró lehet, hogy a négyzetek oldalmérete között lehet egy-egy pixel eltérés. A pályát úgy csináltam, hogy elég sok hely legyen az akadályok (egy kör, egy háromszög és egy hatszög), és a 4 fal között, ahol úgy látszik, hogy a kígyó el fog férni, ott tényleg el is fog. A kígyó fejét jelző háromszög mindig az aktuális menetirányba mutat, de lépni csak bizonyos időközönként lép. Emiatt ha a kígyó 180°-ot akarna fordulni, még a következő ugrásig „vissza lehet vonni” a lépést.

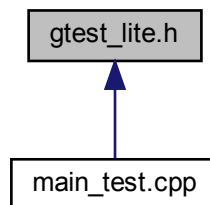
7.12. gtest_lite.h fájlreferencia

```
#include <iostream>
#include <cassert>
#include <cmath>
#include <cstring>
#include <limits>
#include <cstdlib>
#include <string>
#include <fstream>
```

A gtest_lite.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- struct [_Is_Types< F, T >](#)
Segédsablon típuskonverzió futás közbeni ellenőrzésére.
- struct [gtest_lite::Test](#)
- class [gtest_lite::ostreamRedir](#)

Névterek

- [gtest_lite](#)
gtest_lite: a keretrendszer függvényinek és objektumainak névtére

Makródefiníciók

- `#define TEST(C, N) do { gtest_lite::test.begin(#C"##N);`
- `#define END gtest_lite::test.end(); } while (false);`
Tesztet vége.
- `#define ENDM gtest_lite::test.end(true); } while (false);`
- `#define ENDMsg(t) gtest_lite::test.end(true) << t << std::endl; } while (false);`
- `#define SUCCEED() gtest_lite::test.expect(true, __FILE__, __LINE__, "SUCCEED()", true)`
Sikeres teszt makrója.
- `#define FAIL() gtest_lite::test.expect(false, __FILE__, __LINE__, "FAIL()", true)`
Sikertelen teszt fatális hiba makrója.
- `#define ADD_FAILURE() gtest_lite::test.expect(false, __FILE__, __LINE__, "ADD_FAILURE()", true)`
Sikertelen teszt makrója.
- `#define EXPECT_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::eq, __FILE__, __LINE__, "EXPECT_EQ(" #expected ", " #actual ")")`
Azonosságot elváró makró
- `#define EXPECT_NE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::ne, __FILE__, __LINE__, "EXPECT_NE(" #expected ", " #actual ")", "etalon")`
Eltérést elváró makró
- `#define EXPECT_LE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::le, __FILE__, __LINE__, "EXPECT_LE(" #expected ", " #actual ")", "etalon")`
Kisebb, vagy egyenlő relációt elváró makró
- `#define EXPECT_LT(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::lt, __FILE__, __LINE__, "EXPECT_LT(" #expected ", " #actual ")", "etalon")`
Kisebb, mint relációt elváró makró
- `#define EXPECT_GE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::ge, __FILE__, __LINE__, "EXPECT_GE(" #expected ", " #actual ")", "etalon")`
Nagyobb, vagy egyenlő relációt elváró makró
- `#define EXPECT_GT(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::gt, __FILE__, __LINE__, "EXPECT_GT(" #expected ", " #actual ")", "etalon")`
Nagyobb, mint relációt elváró makró
- `#define EXPECT_TRUE(actual) gtest_lite::EXPECT_(true, actual, gtest_lite::eq, __FILE__, __LINE__, "EXPECT_TRUE(" #actual ")")`
Igaz értéket elváró makró
- `#define EXPECT_FALSE(actual) gtest_lite::EXPECT_(false, actual, gtest_lite::eq, __FILE__, __LINE__, "EXPECT_FALSE(" #actual ")")`
Hamis értéket elváró makró
- `#define EXPECT_FLOAT_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, __LINE__, "EXPECT_FLOAT_EQ(" #expected ", " #actual ")")`
Valós számok azonosságát elváró makró
- `#define EXPECT_DOUBLE_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, __LINE__, "EXPECT_DOUBLE_EQ(" #expected ", " #actual ")")`
Valós számok azonosságát elváró makró
- `#define EXPECT_STREQ(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstr, __FILE__, __LINE__, "EXPECT_STREQ(" #expected ", " #actual ")")`
*C stringek (const char *) azonosságát tesztelő makró*
- `#define EXPECT_STRNE(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestr, __FILE__, __LINE__, "EXPECT_STRNE(" #expected ", " #actual ")", "etalon")`
*C stringek (const char *) eltérést tesztelő makró*
- `#define EXPECT_STRCASEEQ(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_STRCASEEQ(" #expected ", " #actual ")")`
*C stringek (const char *) azonosságát tesztelő makró (kisbetű/nagybetű azonos)*

- `#define EXPECT_STRCASENE(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestrcase, __FILE__, __LINE__, "EXPECT_STRCASENE(" #expected ", " #actual ")", "etalon")`
*C stringek (const char *) eltéréset tesztelő makró (kisbetű/nagybetű azonos)*
- `#define EXPECT_THROW(statement, exception_type)`
Kivételt várunk.
- `#define EXPECT_ANY_THROW(statement)`
Kivételt várunk.
- `#define EXPECT_NO_THROW(statement)`
Nem várunk kivételt.
- `#define ASSERT_NO_THROW(statement)`
Nem várunk kivételt.
- `#define EXPECT_THROW_THROW(statement, exception_type)`
Kivételt várunk és továbbdobjuk – ilyen nincs a gtest-ben.
- `#define EXPECT_ENVEQ(expected, actual) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstr, __FILE__, __LINE__, "EXPECT_ENVEQ(" #expected ", " #actual ")", ")`
Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben.
- `#define EXPECT_ENVCASEEQ(expected, actual) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_ENVCASEEQ(" #expected ", " #actual ")", ")`
Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben (kisbetű/nagybetű azonos)
- `#define ASSERT_EQ(expected, actual) gtest_lite::ASSERT_(expected, actual, gtest_lite::eq, "ASSER_EQ")`
Azonosságot elváró makró
- `#define ASSERT_NO_THROW(statement)`
Nem várunk kivételt.
- `#define CREATE_Has_(X)`
- `#define CREATE_Has_fn_(X, S)`
- `#define EXPECTTHROW(statement, exp, act)`
EXPECTTHROW: kivételkezelés.
- `#define ASSERTTHROW(statement, exp, act)`
- `#define ASSERT_(expected, actual, fn, op)`
- `#define GTINIT(is)`
- `#define GTEND(os) os << magic << (gtest_lite::test.fail() ? " NO" : " OK?") << std::endl;`

Függvények

- `void hasMember (...)`
- `template<typename T1, typename T2>
std::ostream & gtest_lite::EXPECT_(T1 exp, T2 act, bool(*pred)(T1, T1), const char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")`
általános sablon a várt értékhez.
- `template<typename T1, typename T2>
std::ostream & gtest_lite::EXPECT_(T1 *exp, T2 *act, bool(*pred)(T1 *, T1 *), const char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")`
pointerre specializált sablon a várt értékhez.
- `std::ostream & gtest_lite::EXPECTSTR(const char *exp, const char *act, bool(*pred)(const char *, const char *), const char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")`
- `template<typename T>
bool gtest_lite::eq(T a, T b)`
- `bool gtest_lite::eqstr(const char *a, const char *b)`
- `bool gtest_lite::eqstrcase(const char *a, const char *b)`
- `template<typename T>
bool gtest_lite::ne(T a, T b)`
- `bool gtest_lite::nestr(const char *a, const char *b)`

- `template<typename T>`
`bool gtest_lite::le (T a, T b)`
- `template<typename T>`
`bool gtest_lite::lt (T a, T b)`
- `template<typename T>`
`bool gtest_lite::ge (T a, T b)`
- `template<typename T>`
`bool gtest_lite::gt (T a, T b)`
- `template<typename T>`
`bool gtest_lite::almostEQ (T a, T b)`

Változók

- `static Test & gtest_lite::test = Test::getTest()`

7.12.1. Részletes leírás

(v4/2022)

Google gtest keretrendszerhez hasonló rendszer. Sz.l. 2015., 2016., 2017. (*Has_X*) Sz.l. 2018 (*template*), *E←* NDM, *ENDMsg*, *nullptr_t* Sz.l. 2019 *singleton* Sz.l. 2021 *ASSERT...*, *STRCASE...* Sz.l. 2021 *EXPECT_REGEX*, *CREATE_Has_fn*, *cmp w. NULL*, *EXPECT_param* fix V.B., Sz.l. 2022 *almostEQ* fix

A tesztelés legalapvetőbb funkcióit támogató függvények és makrók. Nem szálbiztos megvalósítás.

Szabadon felhasználható, bővíthető.

Használati példa: Teszteljük az $f(x)=2*x$ függvényt: `int f(int x) { return 2*x; }`

```
int main() { TEST(TeszEsetNeve, TesztNeve) EXPECT_EQ(0, f(0)); EXPECT_EQ(4, f(2)) << "A függvény hibás
eredményt adott" << std::endl; ... END ... // Fatális hiba esetén a tesztet nem fut tovább. Ezek az A←
SSERT... makrók. // Nem lehet a kiírásukhoz további üzenetet fűzni. PL: TEST(TeszEsetNeve, TesztNeve)
ASSERT_NO_THROW(f(0)); // itt nem lehet << "duma" EXPECT_EQ(4, f(2)) << "A függvény hibás eredményt
adott" << std::endl; ... END ...
```

A működés részleteinek megértése szorgalmi feladat.

7.12.2. Makródefiníciók dokumentációja

7.12.2.1. ADD_FAILURE

```
#define ADD_FAILURE( ) gtest_lite::test.expect(false, __FILE__, __LINE__, "ADD_FAILURE()",
true)
```

Sikertelen teszt makrója.

7.12.2.2. ASSERT_

```
#define ASSERT_(
    expected,
    actual,
    fn,
    op )
```

Érték:

```
EXPECT_(expected, actual, fn, __FILE__, __LINE__, #op "(" #expected " ", " #actual ")"); \
if (!gtest_lite::test.status) { gtest_lite::test.end(); break; }
```

7.12.2.3. ASSERT_EQ

```
#define ASSERT_EQ(
    expected,
    actual ) gtest_lite::ASSERT_(expected, actual, gtest_lite::eq, "ASER_EQ")
```

Azonosságot elváró makró

ASSERT típusú ellenőrzések. Csak 1-2 van megvalósítva. Nem ostream& -val térnek vissza !!! Kivételt várunk

7.12.2.4. ASSERT_NO_THROW [1/2]

```
#define ASSERT_NO_THROW(
    statement )
```

Érték:

```
try { gtest_lite::test.tmp = true; statement; } \
catch (...) { gtest_lite::test.tmp = false; }\
ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.12.2.5. ASSERT_NO_THROW [2/2]

```
#define ASSERT_NO_THROW(
    statement )
```

Érték:

```
try { gtest_lite::test.tmp = true; statement; } \
catch (...) { gtest_lite::test.tmp = false; }\
ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.12.2.6. ASSERTTHROW

```
#define ASSERTTHROW(
    statement,
    exp,
    act )
```

Érték:

```
gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__, __LINE__, #statement) \
« "*** Az utasítás " « (act) \
« "\n** Azt vartuk, hogy " « (exp) « std::endl; if (!gtest_lite::test.status) { gtest_lite::test.end(); \
break; }
```

7.12.2.7. CREATE_Has_

```
#define CREATE_Has_(
    X )
```

Érték:

```
template<typename T> struct _Has_##X { \
    struct Fallback { int X; }; \
    struct Derived : T, Fallback {}; \
    template<typename C, C> struct ChT; \
    template<typename D> static char (&f(ChT<int Fallback::*, &D::X>*)) [1]; \
    template<typename D> static char (&f(...)) [2]; \
    static bool const member = sizeof(f<Derived>(0)) == 2; \
};
```

Segédmakró egy adattag, vagy tagfüggvény létezésének tesztelésére futási időben Ötlet: <https://cpptalk.wordpress.com/2009/09/12/substitution-failure-is-not-an-error-2>

Használat: `CREATE_Has_(size)` ... if (`_Has_size<std::string>::member`)...

7.12.2.8. CREATE_Has_fn_

```
#define CREATE_Has_fn_(
    X,
    S )
```

Érték:

```
template<typename R, typename T> struct _Has_fn_##X##_##S { \
    template<typename C, R (C::*f)() S> struct ChT; \
    template<typename D> static char (&f(ChT<D, &D::X>*)) [1]; \
    template<typename D> static char (&f(...)) [2]; \
    static bool const fn = sizeof(f<T>(0)) == 1; \
};
```

7.12.2.9. END

```
#define END gtest_lite::test.end(); } while (false);
```

Teszteteset vége.

7.12.2.10. ENDM

```
#define ENDM gtest_lite::test.end(true); } while (false);
```

Teszteteset vége allokált blokkok számának összehasonlításával Ez az ellenőrzés nem bomba biztos.

7.12.2.11. ENDMsg

```
#define ENDMsg(
    t ) gtest_lite::test.end(true) << t << std::endl; } while (false);
```

Teszteteset vége allokált blokkok számának összehasonlításával Ez az ellenőrzés nem bomba biztos. Ha hiba van kiírja az üzenetet.

7.12.2.12. EXPECT_ANY_THROW

```
#define EXPECT_ANY_THROW(
    statement )
```

Érték:

```
try { gtest_lite::test.tmp = false; statement; } \
catch (...) { gtest_lite::test.tmp = true; } \
EXPECT_THROW(statement, "kivetelt dob.", "nem dobott kivetelt.")
```

Kivételt várunk.

7.12.2.13. EXPECT_DOUBLE_EQ

```
#define EXPECT_DOUBLE_EQ(
    expected,
```

```

        actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, ↵
__LINE__, "EXPECT_DOUBLE_EQ(" #expected ", " #actual ")") )

```

Valós számok azonosságát elváró makró

7.12.2.14. EXPECT_ENVCASEEQ

```

#define EXPECT_ENVCASEEQ(
    expected,
    actual ) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstrcase,
__FILE__, __LINE__, "EXPECT_ENVCASEEQ(" #expected ", " #actual ")") )

```

Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben (kisbetű/nagybetű azonos)

7.12.2.15. EXPECT_ENVEQ

```

#define EXPECT_ENVEQ(
    expected,
    actual ) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstr,
__FILE__, __LINE__, "EXPECT_ENVEQ(" #expected ", " #actual ")") )

```

Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben.

7.12.2.16. EXPECT_EQ

```

#define EXPECT_EQ(
    expected,
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::eq, __FILE__, __LINE↵
__, "EXPECT_EQ(" #expected ", " #actual ")") )

```

Azonosságot elváró makró

7.12.2.17. EXPECT_FALSE

```

#define EXPECT_FALSE(
    actual ) gtest_lite::EXPECT_(false, actual, gtest_lite::eq, __FILE__, __LINE__,
"EXPECT_FALSE(" #actual ")") )

```

Hamis értéket elváró makró

7.12.2.18. EXPECT_FLOAT_EQ

```

#define EXPECT_FLOAT_EQ(
    expected,
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, ↵
__LINE__, "EXPECT_FLOAT_EQ(" #expected ", " #actual ")") )

```

Valós számok azonosságát elváró makró

7.12.2.19. EXPECT_GE

```

#define EXPECT_GE(
    expected,
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::ge, __FILE__, __LINE↵
__, "EXPECT_GE(" #expected ", " #actual ")", "etalon") )

```

Nagyobb, vagy egyenlő relációt elváró makró

7.12.2.20. EXPECT_GT

```
#define EXPECT_GT(  
    expected,  
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::gt, __FILE__, __LINE__  
    __, "EXPECT_GT(" #expected ", " #actual ")", "etalon" )
```

Nagyobb, mint relációt elváró makró

7.12.2.21. EXPECT_LE

```
#define EXPECT_LE(  
    expected,  
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::le, __FILE__, __LINE__  
    __, "EXPECT_LE(" #expected ", " #actual ")", "etalon" )
```

Kisebb, vagy egyenlő relációt elváró makró

7.12.2.22. EXPECT_LT

```
#define EXPECT_LT(  
    expected,  
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::lt, __FILE__, __LINE__  
    __, "EXPECT_LT(" #expected ", " #actual ")", "etalon" )
```

Kisebb, mint relációt elváró makró

7.12.2.23. EXPECT_NE

```
#define EXPECT_NE(  
    expected,  
    actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::ne, __FILE__, __LINE__  
    __, "EXPECT_NE(" #expected ", " #actual ")", "etalon" )
```

Eltérést elváró makró

7.12.2.24. EXPECT_NO_THROW

```
#define EXPECT_NO_THROW(  
    statement )
```

Érték:

```
try { gtest_lite::test.tmp = true; statement; } \  
catch (...) { gtest_lite::test.tmp = false; }\  
EXPECT_THROW(statement, "nem dob kivételt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.12.2.25. EXPECT_STRCASEEQ

```
#define EXPECT_STRCASEEQ(  
    expected,  
    actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstrcase, __FILE__  
    __, __LINE__, "EXPECT_STRCASEEQ(" #expected ", " #actual ")" )
```

C stringek (const char *) azonosságát tesztelő makró (kisbetű/nagybetű azonos)

7.12.2.26. EXPECT_STRCASENE

```
#define EXPECT_STRCASENE(  
    expected,
```

```

        actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestrcase, __FILE__↵
_, __LINE__, "EXPECT_STRCASENE(" #expected ", " #actual ")", "etalon" )

```

C stringek (const char *) eltéréset tesztelő makró (kisbetű/nagybetű azonos)

7.12.2.27. EXPECT_STREQ

```

#define EXPECT_STREQ(
    expected,
    actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstr, __FILE__, ↵
__LINE__, "EXPECT_STREQ(" #expected ", " #actual " )" )

```

C stringek (const char *) azonosságát tesztelő makró

7.12.2.28. EXPECT_STRNE

```

#define EXPECT_STRNE(
    expected,
    actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestr, __FILE__, ↵
__LINE__, "EXPECT_STRNE(" #expected ", " #actual ")", "etalon" )

```

C stringek (const char *) eltéréset tesztelő makró

7.12.2.29. EXPECT_THROW

```

#define EXPECT_THROW(
    statement,
    exception_type )

```

Érték:

```

try { gtest_lite::test.tmp = false; statement; } \
catch (exception_type) { gtest_lite::test.tmp = true; } \
catch (...) { } \
EXPECT_THROW(statement, "kivetelt dob.", "nem dobott '" #exception_type "' kivetelt.")

```

Kivételt várunk.

7.12.2.30. EXPECT_THROW_THROW

```

#define EXPECT_THROW_THROW(
    statement,
    exception_type )

```

Érték:

```

try { gtest_lite::test.tmp = false; statement; } \
catch (exception_type) { gtest_lite::test.tmp = true; throw; } \
EXPECT_THROW(statement, "kivetelt dob.", "nem dobott '" #exception_type "' kivetelt.")

```

Kivételt várunk és továbbdobjuk – ilyen nincs a gtest-ben.

7.12.2.31. EXPECT_TRUE

```

#define EXPECT_TRUE(
    actual ) gtest_lite::EXPECT_(true, actual, gtest_lite::eq, __FILE__, __LINE__↵
, "EXPECT_TRUE(" #actual " )" )

```

Igaz értéket elváró makró

7.12.2.32. EXPECTTHROW

```

#define EXPECTTHROW(
    statement,
    exp,
    act )

```


Érték:

```
gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__, __LINE__, #statement) \
« "*** Az utasítás " « (act) \
« "\n** Azt vartuk, hogy " « (exp) « std::endl
```

EXPECTTHROW: kivételkezelés.

Belső megvalósításhoz tartozó makrók, és osztályok.

7.12.2.33. Nem célszerű közvetlenül használni, vagy módosítani**7.12.2.34. FAIL**

```
#define FAIL( ) gtest_lite::test.expect(false, __FILE__, __LINE__, "FAIL()", true)
```

Sikertelen teszt fatális hiba makrója.

7.12.2.35. GTEND

```
#define GTEND(
    os ) os << magic << (gtest_lite::test.fail() ? " NO" : " OK?") << std::endl;
```

7.12.2.36. GTINIT

```
#define GTINIT(
    is )
```

Érték:

```
int magic; \
is » magic;
```

7.12.2.37. SUCCEED

```
#define SUCCEED( ) gtest_lite::test.expect(true, __FILE__, __LINE__, "SUCCEED()", true)
```

Sikeres teszt makrója.

7.12.2.38. TEST

```
#define TEST(
    C,
    N ) do { gtest_lite::test.begin(#C".#"N);
```

Teszt kezdete. A makró paraméterezése hasonlít a gtest paraméterezéséhez. Így az itt elkészített tesztek könnyen átemelhetők a gtest keretrendszerbe.

Paraméterek

<i>C</i>	- teszteteset neve (csak a gtest kompatibilitás miatt van külön neve az eseteknek)
<i>N</i>	- teszt neve

7.12.3. Függvények dokumentációja**7.12.3.1. hasMember()**

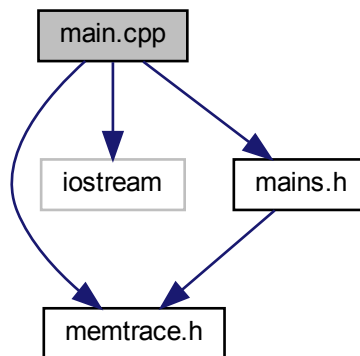
```
void hasMember (
    ... ) [inline]
```

Segédfüggvény egy publikus adattag, vagy tagfüggvény létezésének tesztelésére fordítási időben

7.13. main.cpp fájlreferencia

```
#include "memtrace.h"  
#include <iostream>  
#include "mains.h"
```

A main.cpp definíciós fájl függési gráfja:



Függvények

- int [main](#) ()

7.13.1. Részletes leírás

A főprogram, ami a definiált makróknak megfelelően elindítja a kívánt maineket.

7.13.2. Függvények dokumentációja

7.13.2.1. main()

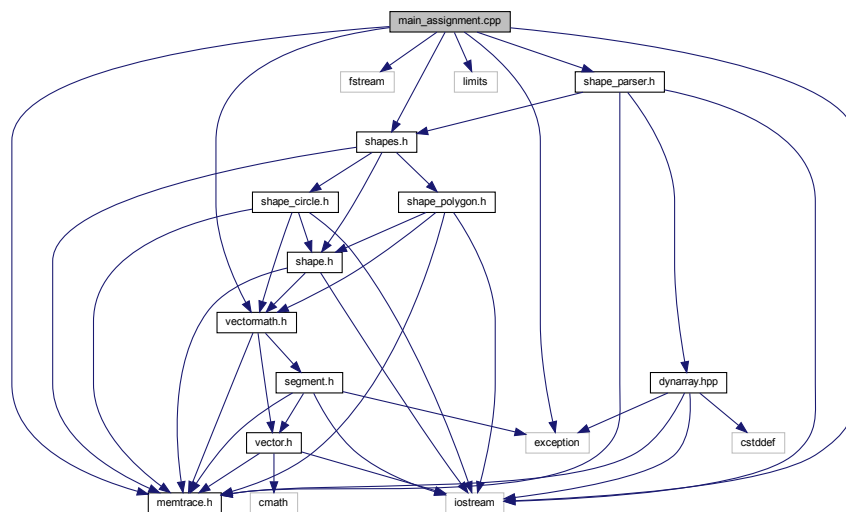
```
int main ( )
```

A főprogram, ami a definiált makróknak megfelelően elindítja a kívánt maineket.

7.14. main_assignment.cpp fájlreferencia

```
#include "memtrace.h"  
#include <iostream>  
#include <fstream>  
#include <exception>  
#include <limits>  
#include "vectormath.h"  
#include "shapes.h"  
#include "shape_parser.h"
```

A main_assignment.cpp definíciós fájl függési gráfja:



Névterek

- [anonymous_namespace{main_assignment.cpp}](#)

Függvények

- bool [anonymous_namespace{main_assignment.cpp}::outside_of_unit_circle](#) (const [Shape](#) &s)
- void [main_assignment](#) (std::istream &is, std::ostream &os)

7.14.1. Részletes leírás

A feladat által kért főprogram megvalósítása.

7.14.2. Függvények dokumentációja

7.14.2.1. main_assignment()

```
void main_assignment (
    std::istream & is,
    std::ostream & os )
```

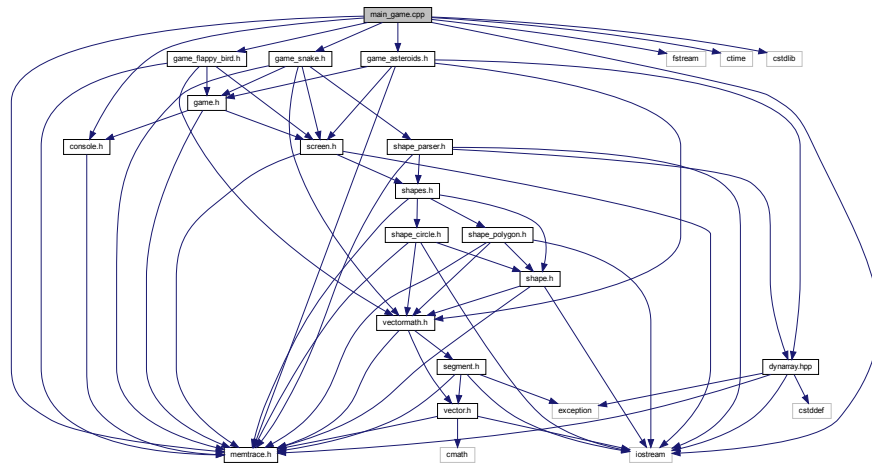
A feladat által kért főprogram megvalósítása.

A bemenetről koordinátapárokat vesz be, és a kimenetre kiírja, hogy a beolvasott fájlból mely alakzatok tartalmazzák azt a pontot. A beolvasáskor eldobja az egységkörrel érintkező alakzatokat. Paramétereket a könnyebb tesztelhetőség miatt kap.

7.15. main_game.cpp fájlreferencia

```
#include "memtrace.h"
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#include "console.h"
```

```
#include "game_flappy_bird.h"
#include "game_snake.h"
#include "game_asteroids.h"
A main_game.cpp definíciós fájl függési gráfja:
```



Függvények

- void `main_game` ()

7.15.1. Részletes leírás

A játékprogramok futtatásáért felelős főprogram.
Csak akkor fordul bele, ha a MAIN_GAME makró definiált.

7.15.2. Függvények dokumentációja

7.15.2.1. main_game()

```
void main_game ( )
```

A játékprogramok futtatásáért felelős főprogram.

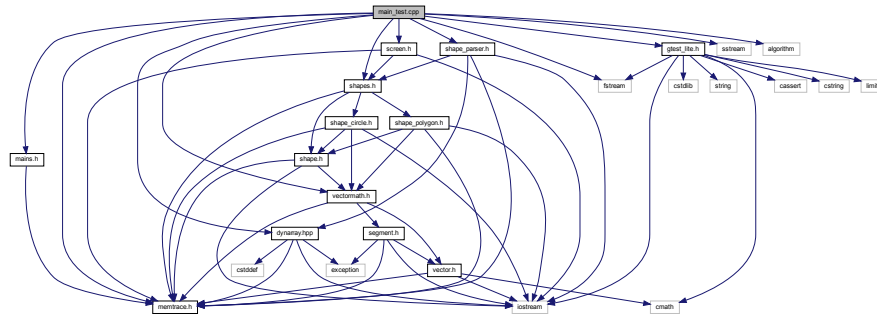
A 3 játék közül véletlenszerűen választ minden alkalommal, és minden játék addig tart, amíg a játékos meghal, vagy 20s letelik. Ha az idő letelt, akkor új játék indul. Meghaláskor a játszott játékok számát kiírja. A kilépés (q gomb) meghalásnak számít.

7.16. main_test.cpp fájlreferencia

```
#include "memtrace.h"
#include <fstream>
#include <sstream>
#include <algorithm>
#include "vectormath.h"
#include "shapes.h"
#include "dynarray.hpp"
#include "shape_parser.h"
#include "screen.h"
#include "mains.h"
```

```
#include "gtest_lite.h"
```

A main_test.cpp definíciós fájl függési gráfja:



Függvények

- void `main_test` ()

7.16.1. Részletes leírás

A tesztprogram, főprogram.

Itt valósulnak meg a unittesztek, illetve a feladat főprogram tesztje is.

7.16.2. Függvények dokumentációja

7.16.2.1. main_test()

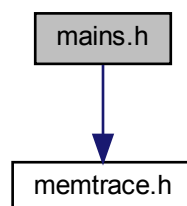
```
void main_test ( )
```

Itt valósulnak meg a unittesztek, illetve a feladat főprogram tesztje is. A játékokat, a játék főprogramot, és a [Console](#) osztályt nem tesztelem, ezeket feltételes fordítással fordítom csak bele a programba, így elkerülve a rossz coverage arányt.

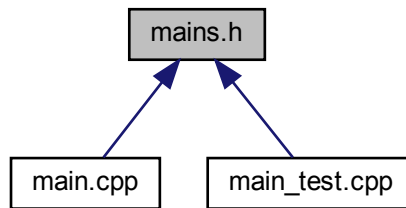
7.17. mains.h fájlreferencia

```
#include "memtrace.h"
```

A mains.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Függvények

- void [main_test](#) ()
- void [main_assignment](#) (std::istream &is, std::ostream &os)
- void [main_game](#) ()

7.17.1. Részletes leírás

A főprogramok deklarációja, túlzásnak éreztem volna ezekre 3 külön fejlécfájlt készíteni.

7.17.2. Függvények dokumentációja

7.17.2.1. main_assignment()

```
void main_assignment (
    std::istream & is,
    std::ostream & os )
```

A feladat által kért főprogram megvalósítása.

A bemenetről koordinátapárokat vesz be, és a kimenetre kiírja, hogy a beolvasott fájlból mely alakzatok tartalmazzák azt a pontot. A beolvasáskor eldobja az egységkörrel érintkező alakzatokat. Paramétereket a könnyebb tesztelhetőség miatt kap.

7.17.2.2. main_game()

```
void main_game ( )
```

A játékprogramok futtatásáért felelős főprogram.

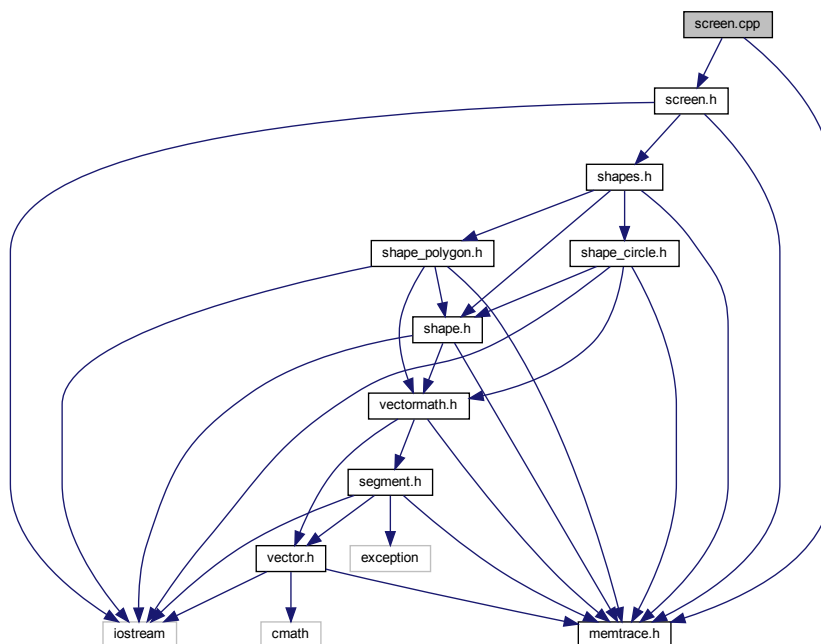
A 3 játék közül véletlenszerűen választ minden alkalommal, és minden játék addig tart, amíg a játékos meghal, vagy 20s letelik. Ha az idő letelt, akkor új játék indul. Meghaláskor a játszott játékok számát kiírja. A kilépés (q gomb) meghalásnak számít.

7.17.2.3. main_test()

```
void main_test ( )
```

Itt valósulnak meg a unittesztek, illetve a feladat főprogram tesztje is. A játékokat, a játék főprogramot, és a [Console](#) osztályt nem teszteltem, ezeket feltételes fordítással fordítom csak bele a programba, így elkerülve a rossz coverage arányt.

A screen.cpp definíciós fájl függési gráfja:



Névterek

- `anonymous_namespace{screen.cpp}`

Enumerációk

- `enum anonymous_namespace{screen.cpp}::Block { anonymous_namespace{screen.cpp}::BLOCK_EMPTY, anonymous_namespace{screen.cpp}::BLOCK_DOWN, anonymous_namespace{screen.cpp}::BLOCK_UP, anonymous_namespace{screen.cpp}::BLOCK_FULL }`

Függvények

- `const char * anonymous_namespace{screen.cpp}::getblock (Block b)`

ad egy dobozt: szökőz, alsó, felső, vagy teljes

7.20.1. Részletes leírás

Alakzatkirajzoló osztály logikája, itt van a dobozok karakterkódolása is elrejtve (makró: CMDEXE_ENCODING).

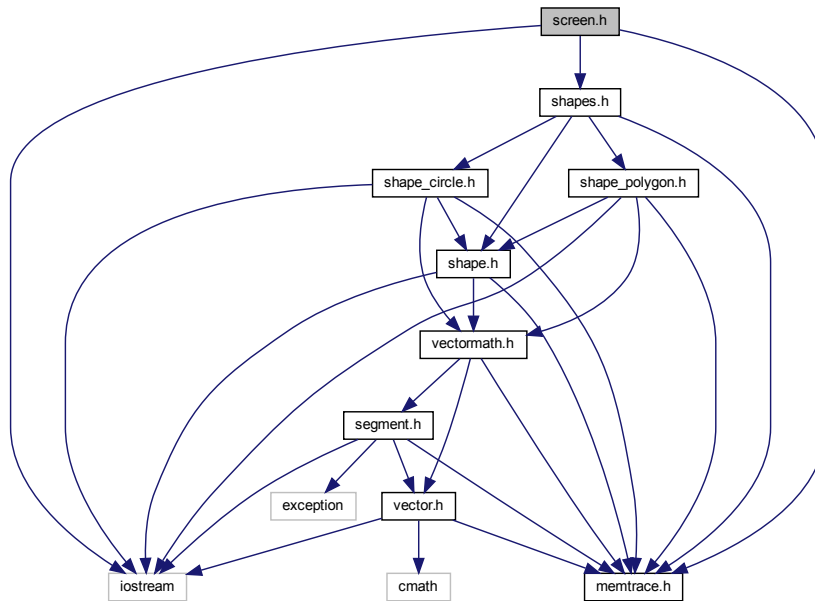
7.21. screen.h fájlreferencia

```

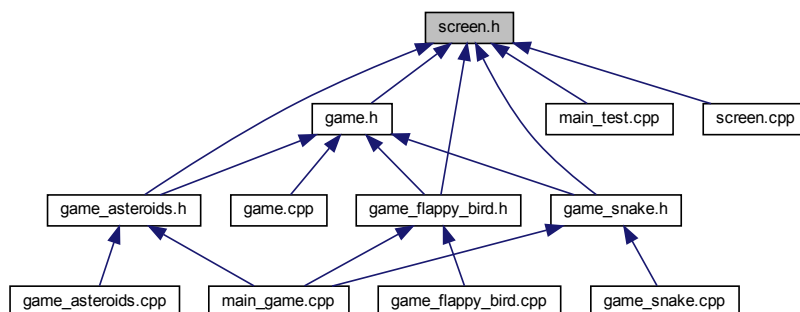
#include "memtrace.h"
#include <iostream>
#include "shapes.h"

```


A screen.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [Screen](#)

Függvények

- `std::ostream & operator<< (std::ostream &out, const Screen &s)`

7.21.1. Részletes leírás

Az alakzatok lerendereléséért felelős osztály.

Az alakzatok `has_point` függvényét hívogatva készít egy képet, amit képes kiírni egy kimeneti folyamra.

7.21.2. Függvények dokumentációja

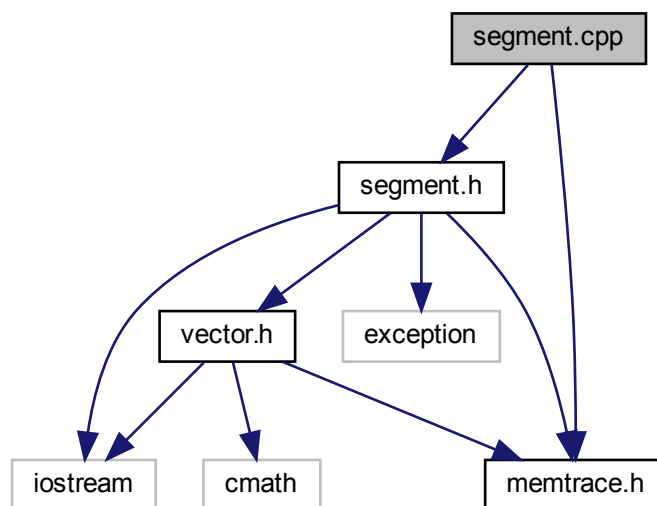
7.21.2.1. operator<<()

```
std::ostream& operator<< (  
    std::ostream & out,  
    const Screen & s ) [inline]
```

7.22. segment.cpp fájlreferencia

```
#include "memtrace.h"  
#include "segment.h"
```

A segment.cpp definíciós fájl függési gráfja:



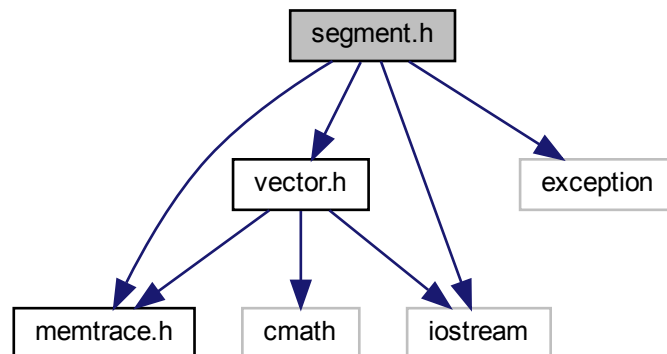
7.22.1. Részletes leírás

A szakasz tagfüggvényei, síkgeometria / koordinátageometria.

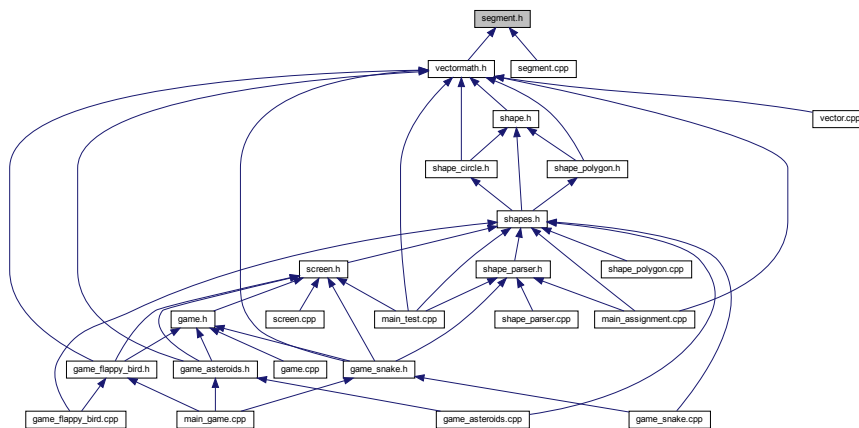
7.23. segment.h fájlreferencia

```
#include "memtrace.h"  
#include <iostream>  
#include <exception>  
#include "vector.h"
```

A segment.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- struct [Segment](#)
egy szakaszt, vagy egyenest reprezentál

Függvények

- `std::ostream & operator<< (std::ostream &out, const Segment &s)`

7.23.1. Részletes leírás

A szakasz, ami izomorf 2db 2 dimenziós ponttal, megvalósítása. Hasznos a sokszög oldalához; ahhoz, hogy egy pont az oldalon belül van-e, illetve a körrel való metszés meghatározására.

7.23.2. Függvények dokumentációja

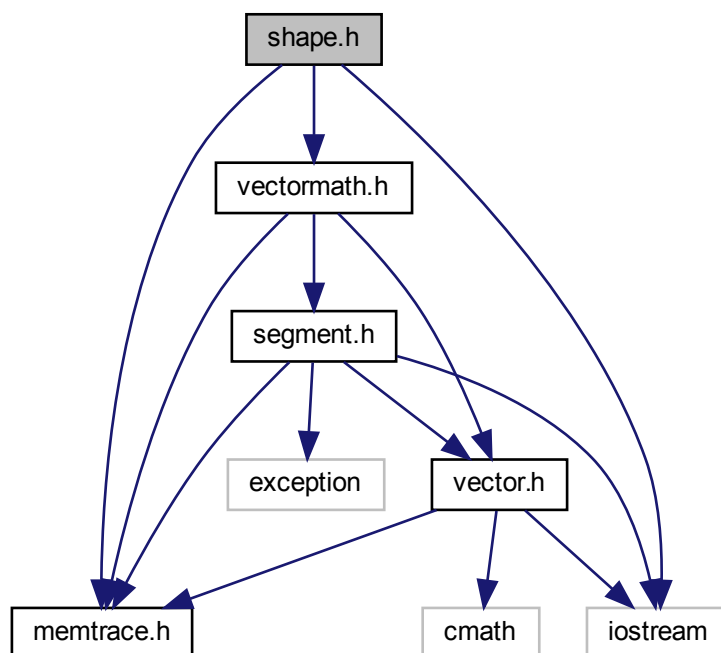
7.23.2.1. operator<<()

```
std::ostream& operator<< (
    std::ostream & out,
    const Segment & s ) [inline]
```

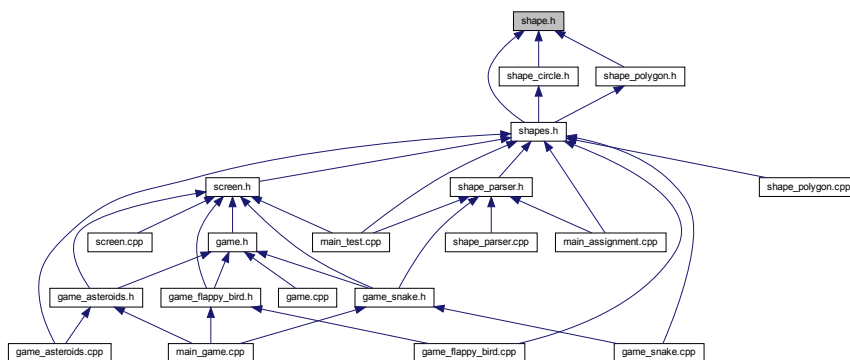
7.24. shape.h fájlreferencia

```
#include "memtrace.h"
#include <iostream>
#include "vectormath.h"
```

A shape.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [Shape](#)

Absztrakt síkidom őssztály.

Függvények

- std::istream & [operator>>](#) (std::istream &is, [Shape](#) &s)
- std::ostream & [operator<<](#) (std::ostream &os, const [Shape](#) &s)

7.24.1. Függvények dokumentációja

7.24.1.1. operator<<()

```
std::ostream& operator<< (  
    std::ostream & os,  
    const Shape & s ) [inline]
```

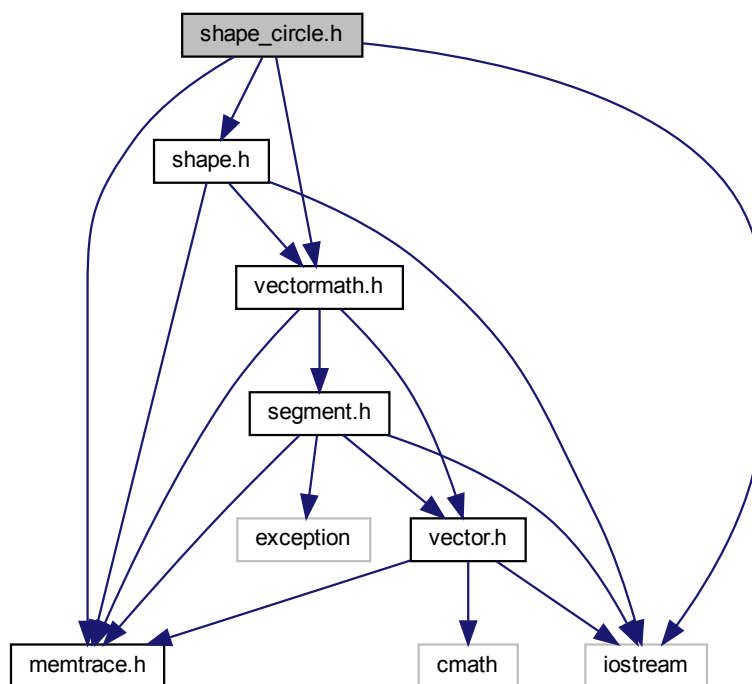
7.24.1.2. operator>>()

```
std::istream& operator>> (  
    std::istream & is,  
    Shape & s ) [inline]
```

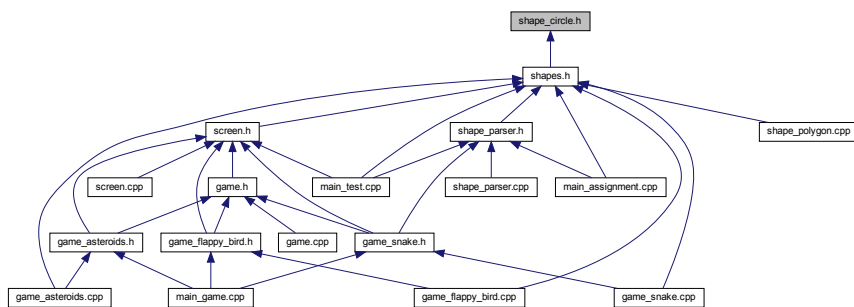
7.25. shape_circle.h fájlreferencia

```
#include "memtrace.h"  
#include <iostream>  
#include "vectormath.h"  
#include "shape.h"
```

A `shape_circle.h` definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class [Circle](#)

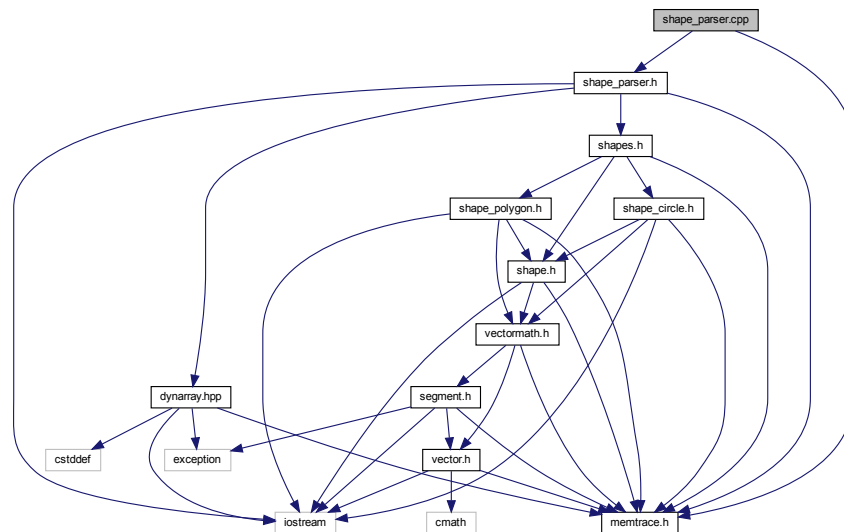
7.25.1. Részletes leírás

A kör alakzatot megvalósító osztály fájlja.

7.26. shape_parser.cpp fájlreferencia

```
#include "memtrace.h"
#include "shape_parser.h"
```

A shape_parser.cpp definíciós fájl függési gráfja:



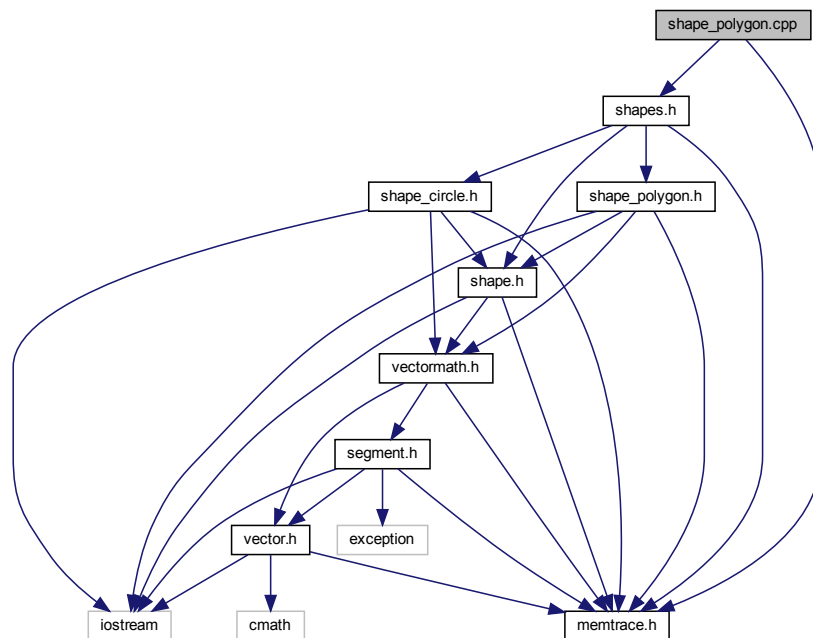
7.26.1. Részletes leírás

Az alakzatok beolvasása és newzása, illetve törlése.

7.27. shape_parser.h fájlreferencia

```
#include "memtrace.h"
#include <iostream>
#include "dynarray.hpp"
#include "shapes.h"
```


A shape_polygon.cpp definíciós fájl függési gráfja:



7.28.1. Részletes leírás

A szabályos sokszög bonyolultabb tagfüggvényei találhatóak itt.

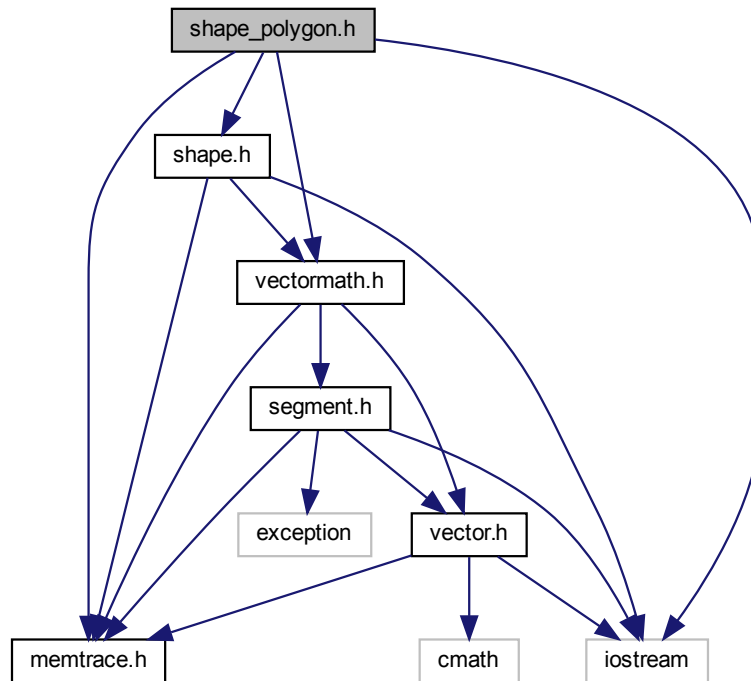
7.29. shape_polygon.h fájlreferencia

```

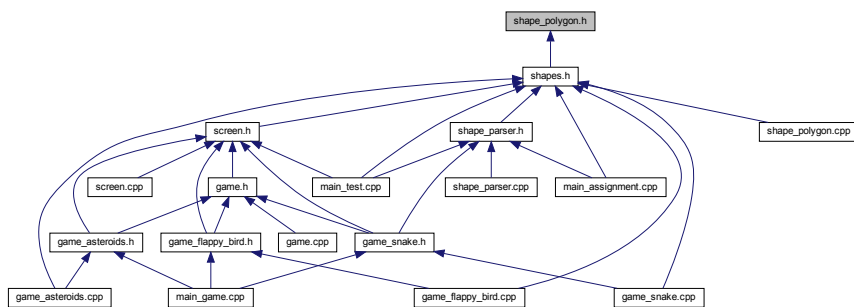
#include "memtrace.h"
#include <iostream>
#include "vectormath.h"
#include "shape.h"

```

A shape_polygon.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

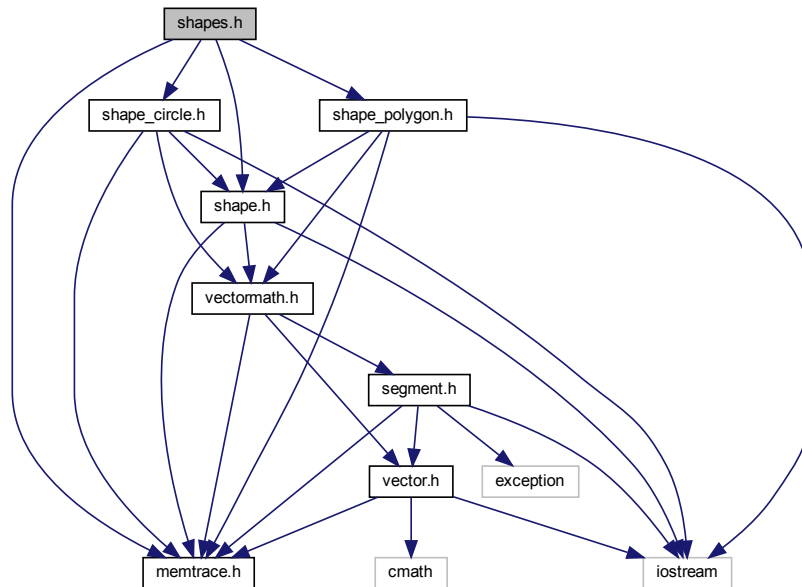
- class [Polygon](#)

7.29.1. Részletes leírás

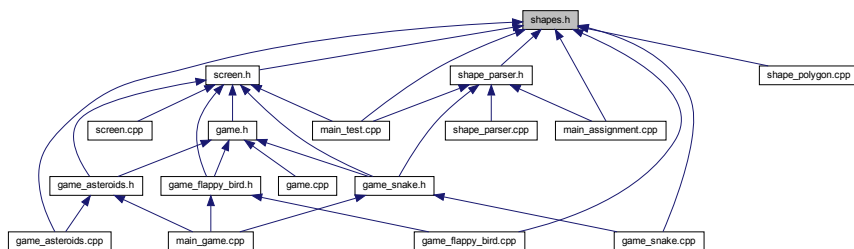
A szabályos sokszöget megvalósító osztály fájlja.

7.30. shapes.h fájlreferencia

```
#include "memtrace.h"
#include "shape.h"
#include "shape_circle.h"
#include "shape_polygon.h"
A shapes.h definíciós fájl függési gráfja:
```



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



7.30.1. Részletes leírás

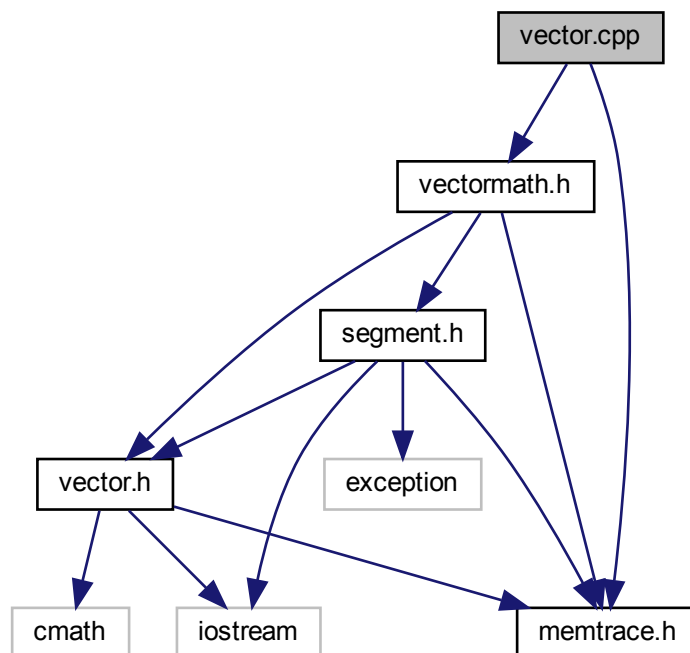
Az eredeti, bontatlan fájlban az összes [Shape](#) itt volt, és gondoltam, meghagyom a fájlt, így 3 include helyett elég 1, és nem kell módosítani a többi fájl emiatt.

7.31. snake_level.txt fájlreferencia

7.32. vector.cpp fájlreferencia

```
#include "memtrace.h"
#include "vectormath.h"
```

A vector.cpp definíciós fájl függési gráfja:



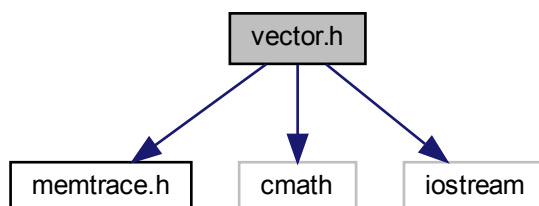
7.32.1. Részletes leírás

Kellett csinálni külön fájlt a statikus konstans adattagoknak.

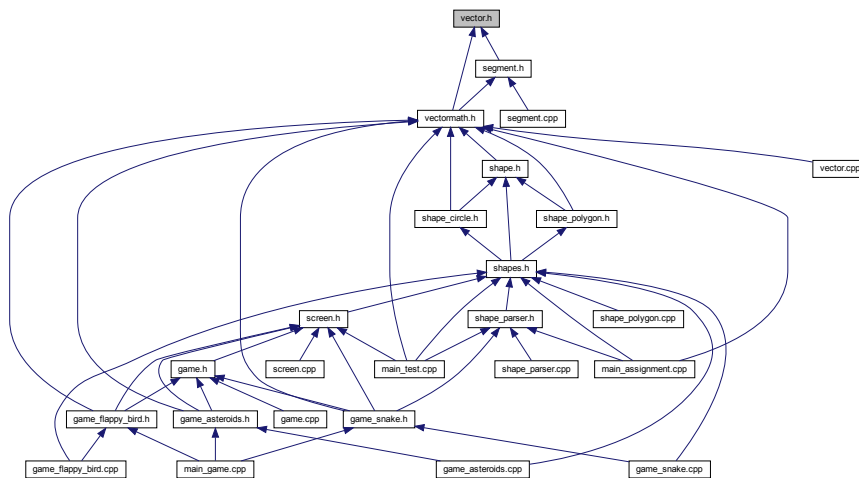
7.33. vector.h fájlreferencia

```
#include "memtrace.h"
#include <cmath>
#include <iostream>
```

A vector.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- struct [Vector](#)

Függvények

- std::istream & [operator>>](#) (std::istream &in, [Vector](#) &v)
- std::ostream & [operator<<](#) (std::ostream &out, const [Vector](#) &v)

7.33.1. Részletes leírás

Két dimenziós, lebegőpontos (double) vektor (= irányított szakasz) megvalósítása.

7.33.2. Függvények dokumentációja

7.33.2.1. operator<<()

```
std::ostream& operator<< (
    std::ostream & out,
    const Vector & v ) [inline]
```

7.33.2.2. operator>>()

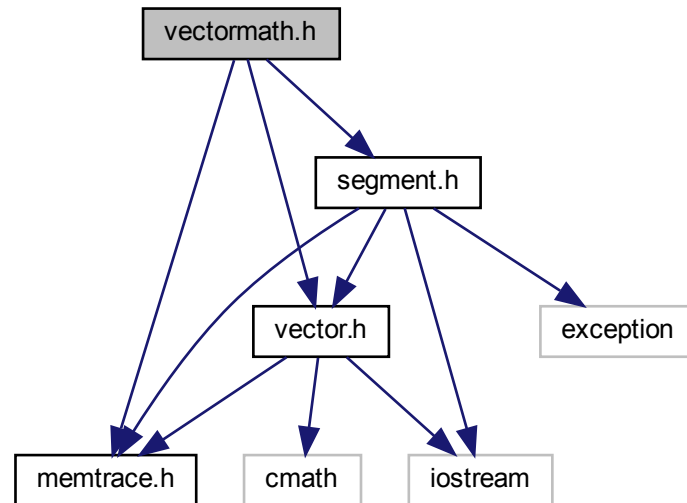
```
std::istream& operator>> (
    std::istream & in,
    Vector & v ) [inline]
```

7.34. vectormath.h fájlreferencia

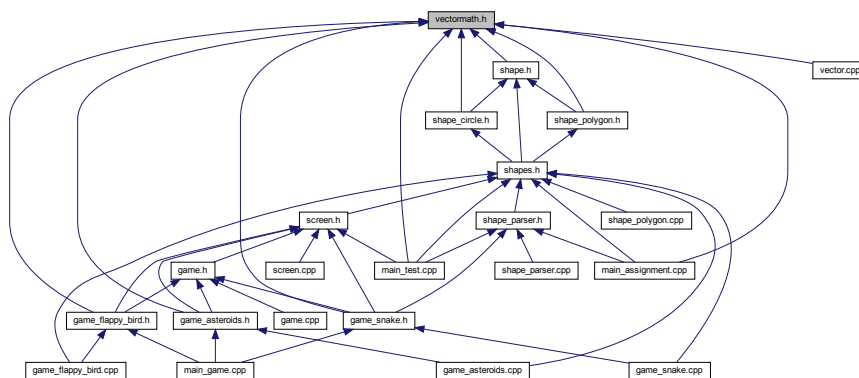
```
#include "memtrace.h"
#include "vector.h"
```

```
#include "segment.h"
```

A vectormath.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Változók

- constexpr double `MATH_PI` = 3.14159265358979323846

7.34.1. Részletes leírás

Lehet include-olni a `PI`-ért, illetve a szétbontás előtt itt voltak a `Vector` és a `Segment` osztályok.

7.34.2. Változók dokumentációja

7.34.2.1. MATH_PI

```
constexpr double MATH_PI = 3.14159265358979323846  [constexpr]
```


Tárgymutató

- `_Is_Types< F, T >`, 15
 - `convertable`, 16
 - `f`, 16
- `~Console`
 - `Console`, 26
- `~DynArray`
 - `DynArray< T >`, 32
- `~Game`
 - `Game`, 35
- `~Screen`
 - `Screen`, 66
- `~Shape`
 - `Shape`, 72
- `~ShapeParser`
 - `ShapeParser`, 76
- `~Test`
 - `gtest_lite::Test`, 78
- `~ostreamRedir`
 - `gtest_lite::ostreamRedir`, 59
- `a`
 - `Segment`, 70
- `ablocks`
 - `gtest_lite::Test`, 80
- `acceleration`
 - `GameAsteroids`, 41
- `Actor`
 - `GameAsteroids::Actor`, 18
- `ADD_FAILURE`
 - `gtest_lite.h`, 101
- `almostEQ`
 - `gtest_lite`, 12
- `angle_to`
 - `Vector`, 83
- `anonymous_namespace{console.cpp}`, 9
- `anonymous_namespace{game_asteroids.cpp}`, 9
 - `randd`, 9
- `anonymous_namespace{main_assignment.cpp}`, 9
 - `outside_of_unit_circle`, 10
- `anonymous_namespace{screen.cpp}`, 10
 - `Block`, 10
 - `getblock`, 10
- `anonymous_namespace{screen.cpp}`
 - `BLOCK_DOWN`, 10
 - `BLOCK_EMPTY`, 10
 - `BLOCK_FULL`, 10
 - `BLOCK_UP`, 10
- `append`
 - `DynArray< T >`, 33
- `array`
 - `ShapeParser`, 76
- `ASSERT_`
 - `gtest_lite.h`, 101
- `ASSERT_EQ`
 - `gtest_lite.h`, 102
- `ASSERT_NO_THROW`
 - `gtest_lite.h`, 102
- `ASSERTTHROW`
 - `gtest_lite.h`, 102
- `astatus`
 - `gtest_lite::Test`, 79
- `asteroids`
 - `GameAsteroids`, 41
- `b`
 - `Segment`, 71
- `begin`
 - `DynArray< T >`, 33
 - `gtest_lite::Test`, 79
 - `ShapeParser`, 76
- `bird_pos`
 - `GameFlappyBird`, 45
- `bird_velocity`
 - `GameFlappyBird`, 45
- `Block`
 - `anonymous_namespace{screen.cpp}`, 10
- `BLOCK_DOWN`
 - `anonymous_namespace{screen.cpp}`, 10
- `BLOCK_EMPTY`
 - `anonymous_namespace{screen.cpp}`, 10
- `BLOCK_FULL`
 - `anonymous_namespace{screen.cpp}`, 10
- `block_size`
 - `GameSnake`, 50
- `BLOCK_UP`
 - `anonymous_namespace{screen.cpp}`, 10
- `C`
 - `console.cpp`, 88
- `center`
 - `Circle`, 23
 - `Polygon`, 64
- `Circle`, 20
 - `center`, 23
 - `Circle`, 22
 - `get_center`, 22
 - `has_point`, 22
 - `intersects_with`, 23

- print_to, 23
- radius, 24
- read_from, 23
- clear
 - Screen, 67
- closest_edge_to
 - Polygon, 62
- closest_point_to
 - Segment, 70
- clrscr
 - Console, 26
- code
 - Console::keyCodes, 58
- con
 - Console, 26
- Console, 24
 - ~Console, 26
 - clrscr, 26
 - con, 26
 - Console, 26
 - getch, 26
 - getsize, 27
 - gotoxy, 27
 - hMeter, 27
 - kbhit, 28
 - KEY_DOWN, 29
 - KEY_HOME, 29
 - KEY_LEFT, 29
 - KEY_RIGHT, 29
 - KEY_UP, 29
 - operator=, 28
 - trCode, 28
- console.cpp, 87
 - C, 88
- console.h, 88
- Console::keyCodes, 57
 - code, 58
 - key, 58
- convertable
 - _Is_Types< F, T >, 16
- CREATE_Has_
 - gtest_lite.h, 102
- CREATE_Has_fn_
 - gtest_lite.h, 103
- current
 - DynArray< T >::Iterator, 57
- data
 - DynArray< T >, 33
 - Screen, 68
- DEFAULT_SIZE
 - DynArray< T >, 33
- delta_since_last_jump
 - GameSnake, 50
- difference_type
 - DynArray< T >::Iterator, 55
- distance_squared_to
 - Vector, 84
- distance_to
 - Vector, 84
- DOWN
 - Vector, 86
- draw_shape
 - Screen, 67
- DynArray
 - DynArray< T >, 32
- DynArray< T >, 30
 - ~DynArray, 32
 - append, 33
 - begin, 33
 - data, 33
 - DEFAULT_SIZE, 33
 - DynArray, 32
 - end, 33
 - length, 34
 - operator=, 33
 - size, 34
- DynArray< T >::Iterator, 54
 - current, 57
 - difference_type, 55
 - end, 57
 - Iterator, 56
 - iterator_category, 55
 - operator!=, 56
 - operator*, 56
 - operator++, 56
 - pointer, 55
 - reference, 56
 - value_type, 56
- dynarray.hpp, 89
- END
 - gtest_lite.h, 103
- end
 - DynArray< T >, 33
 - DynArray< T >::Iterator, 57
 - gtest_lite::Test, 79
 - ShapeParser, 76
- ENDM
 - gtest_lite.h, 103
- ENDMsg
 - gtest_lite.h, 103
- eq
 - gtest_lite, 12
- eqstr
 - gtest_lite, 12
- eqstrcase
 - gtest_lite, 12
- expect
 - gtest_lite::Test, 79
- EXPECT_
 - gtest_lite, 12
- EXPECT_ANY_THROW
 - gtest_lite.h, 103
- EXPECT_DOUBLE_EQ
 - gtest_lite.h, 103
- EXPECT_ENVCASEEQ
 - gtest_lite.h, 104

- EXPECT_ENVEQ
 - gtest_lite.h, [104](#)
- EXPECT_EQ
 - gtest_lite.h, [104](#)
- EXPECT_FALSE
 - gtest_lite.h, [104](#)
- EXPECT_FLOAT_EQ
 - gtest_lite.h, [104](#)
- EXPECT_GE
 - gtest_lite.h, [104](#)
- EXPECT_GT
 - gtest_lite.h, [104](#)
- EXPECT_LE
 - gtest_lite.h, [105](#)
- EXPECT_LT
 - gtest_lite.h, [105](#)
- EXPECT_NE
 - gtest_lite.h, [105](#)
- EXPECT_NO_THROW
 - gtest_lite.h, [105](#)
- EXPECT_STRCASEEQ
 - gtest_lite.h, [105](#)
- EXPECT_STRCASENE
 - gtest_lite.h, [105](#)
- EXPECT_STREQ
 - gtest_lite.h, [106](#)
- EXPECT_STRNE
 - gtest_lite.h, [106](#)
- EXPECT_THROW
 - gtest_lite.h, [106](#)
- EXPECT_THROW_THROW
 - gtest_lite.h, [106](#)
- EXPECT_TRUE
 - gtest_lite.h, [106](#)
- EXPECTSTR
 - gtest_lite, [13](#)
- EXPECTTHROW
 - gtest_lite.h, [106](#)
- f
 - _Is_Types< F, T >, [16](#)
- FAIL
 - gtest_lite.h, [107](#)
- fail
 - gtest_lite::Test, [79](#)
- failed
 - gtest_lite::Test, [80](#)
- forward
 - GameSnake, [50](#)
- Game, [34](#)
 - ~Game, [35](#)
 - input, [35](#)
 - play, [36](#)
 - update, [36](#)
- game.cpp, [90](#)
 - max_delta, [90](#)
- game.h, [91](#)
- game_asteroids.cpp, [92](#)
- game_asteroids.h, [93](#)
- game_flappy_bird.cpp, [94](#)
- game_flappy_bird.h, [94](#)
- game_snake.cpp, [96](#)
- game_snake.h, [96](#)
- GameAsteroids, [37](#)
 - acceleration, [41](#)
 - asteroids, [41](#)
 - GameAsteroids, [39](#)
 - input, [39](#)
 - player, [41](#)
 - seconds_since_asteroid, [41](#)
 - seconds_until_asteroid, [41](#)
 - spawn_random_asteroid, [39](#)
 - update, [40](#)
- GameAsteroids::Actor, [17](#)
 - Actor, [18](#)
 - pos, [19](#)
 - rot, [19](#)
 - size, [19](#)
 - speed, [19](#)
 - update, [18](#)
- GameFlappyBird, [42](#)
 - bird_pos, [45](#)
 - bird_velocity, [45](#)
 - GameFlappyBird, [44](#)
 - gravity, [45](#)
 - input, [44](#)
 - jump_speed, [45](#)
 - lower_wall, [45](#)
 - randomize_spike_height, [44](#)
 - spike_x, [45](#)
 - spike_x_speed, [46](#)
 - spike_y_offset, [46](#)
 - update, [44](#)
 - upper_wall, [46](#)
- GameSnake, [46](#)
 - block_size, [50](#)
 - delta_since_last_jump, [50](#)
 - forward, [50](#)
 - GameSnake, [49](#)
 - input, [49](#)
 - seconds_till_snake_jumps, [50](#)
 - shape_parser, [50](#)
 - SNAKE_LENGTH, [50](#)
 - snake_pos, [50](#)
 - update, [49](#)
- ge
 - gtest_lite, [13](#)
- get_center
 - Circle, [22](#)
- getblock
 - anonymous_namespace{screen.cpp}, [10](#)
- getch
 - Console, [26](#)
- getsize
 - Console, [27](#)
- getTest

- gtest_lite::Test, [80](#)
- gotoxy
 - Console, [27](#)
- gravity
 - GameFlappyBird, [45](#)
- gt
 - gtest_lite, [13](#)
- GTEND
 - gtest_lite.h, [107](#)
- gtest_lite, [11](#)
 - almostEQ, [12](#)
 - eq, [12](#)
 - eqstr, [12](#)
 - eqstrcase, [12](#)
 - EXPECT_, [12](#)
 - EXPECTSTR, [13](#)
 - ge, [13](#)
 - gt, [13](#)
 - le, [13](#)
 - lt, [14](#)
 - ne, [14](#)
 - nestr, [14](#)
 - test, [14](#)
- gtest_lite.h, [98](#)
 - ADD_FAILURE, [101](#)
 - ASSERT_, [101](#)
 - ASSERT_EQ, [102](#)
 - ASSERT_NO_THROW, [102](#)
 - ASSERTTHROW, [102](#)
 - CREATE_Has_, [102](#)
 - CREATE_Has_fn_, [103](#)
 - END, [103](#)
 - ENDM, [103](#)
 - ENDMsg, [103](#)
 - EXPECT_ANY_THROW, [103](#)
 - EXPECT_DOUBLE_EQ, [103](#)
 - EXPECT_ENVCASEEQ, [104](#)
 - EXPECT_ENVEQ, [104](#)
 - EXPECT_EQ, [104](#)
 - EXPECT_FALSE, [104](#)
 - EXPECT_FLOAT_EQ, [104](#)
 - EXPECT_GE, [104](#)
 - EXPECT_GT, [104](#)
 - EXPECT_LE, [105](#)
 - EXPECT_LT, [105](#)
 - EXPECT_NE, [105](#)
 - EXPECT_NO_THROW, [105](#)
 - EXPECT_STRCASEEQ, [105](#)
 - EXPECT_STRCASENE, [105](#)
 - EXPECT_STREQ, [106](#)
 - EXPECT_STRNE, [106](#)
 - EXPECT_THROW, [106](#)
 - EXPECT_THROW_THROW, [106](#)
 - EXPECT_TRUE, [106](#)
 - EXPECTTHROW, [106](#)
 - FAIL, [107](#)
 - GTEND, [107](#)
 - GTINIT, [107](#)
 - hasMember, [107](#)
 - SUCCEED, [107](#)
 - TEST, [107](#)
- gtest_lite::ostreamRedir, [58](#)
 - ~ostreamRedir, [59](#)
 - ostreamRedir, [59](#)
 - save, [59](#)
 - src, [59](#)
- gtest_lite::Test, [77](#)
 - ~Test, [78](#)
 - ablocks, [80](#)
 - astatus, [79](#)
 - begin, [79](#)
 - end, [79](#)
 - expect, [79](#)
 - fail, [79](#)
 - failed, [80](#)
 - getTest, [80](#)
 - name, [80](#)
 - null, [80](#)
 - operator=, [80](#)
 - os, [81](#)
 - status, [81](#)
 - sum, [81](#)
 - Test, [78](#)
 - tmp, [81](#)
- GTINIT
 - gtest_lite.h, [107](#)
- has_point
 - Circle, [22](#)
 - Polygon, [63](#)
 - Shape, [72](#)
- hasMember
 - gtest_lite.h, [107](#)
- height
 - Screen, [68](#)
- hMeter
 - Console, [27](#)
- idx
 - Screen, [67](#)
- input
 - Game, [35](#)
 - GameAsteroids, [39](#)
 - GameFlappyBird, [44](#)
 - GameSnake, [49](#)
- intersects_with
 - Circle, [23](#)
 - Polygon, [63](#)
 - Shape, [73](#)
- is_point_to_the_left
 - Segment, [70](#)
- Iterator
 - DynArray< T >::Iterator, [56](#)
 - ShapeParser::Iterator, [53](#)
- iterator_category
 - DynArray< T >::Iterator, [55](#)

jump_speed
 GameFlappyBird, 45

kbhit
 Console, 28

key
 Console::keyCodes, 58

KEY_DOWN
 Console, 29

KEY_HOME
 Console, 29

KEY_LEFT
 Console, 29

KEY_RIGHT
 Console, 29

KEY_UP
 Console, 29

le
 gtest_lite, 13

LEFT
 Vector, 86

length
 DynArray< T >, 34

lower_wall
 GameFlappyBird, 45

lt
 gtest_lite, 14

main
 main.cpp, 108

main.cpp, 108
 main, 108

main_assignment
 main_assignment.cpp, 109
 mains.h, 112

main_assignment.cpp, 108
 main_assignment, 109

main_game
 main_game.cpp, 110
 mains.h, 112

main_game.cpp, 109
 main_game, 110

main_test
 main_test.cpp, 111
 mains.h, 112

main_test.cpp, 110
 main_test, 111

mains.h, 111
 main_assignment, 112
 main_game, 112
 main_test, 112

MATH_PI
 vectormath.h, 128

max_delta
 game.cpp, 90

memtrace.cpp, 113

memtrace.h, 113

name
 gtest_lite::Test, 80

ne
 gtest_lite, 14

nestr
 gtest_lite, 14

null
 gtest_lite::Test, 80

operator!=
 DynArray< T >::Iterator, 56

operator<<
 screen.h, 116
 segment.h, 117
 shape.h, 119
 vector.h, 127

operator>>
 shape.h, 119
 vector.h, 127

operator*
 DynArray< T >::Iterator, 56
 ShapeParser::Iterator, 53
 Vector, 84

operator+
 Vector, 84

operator++
 DynArray< T >::Iterator, 56

operator+=
 Vector, 84

operator-
 Vector, 85

operator=
 Console, 28
 DynArray< T >, 33
 gtest_lite::Test, 80
 Screen, 67
 ShapeParser, 76

os
 gtest_lite::Test, 81

ostreamRedir
 gtest_lite::ostreamRedir, 59

outside_of_unit_circle
 anonymous_namespace{main_assignment.cpp},
 10

play
 Game, 36

player
 GameAsteroids, 41

pointer
 DynArray< T >::Iterator, 55

polar
 Vector, 85

Polygon, 60
 center, 64
 closest_edge_to, 62
 has_point, 63
 intersects_with, 63
 Polygon, 62

- print_to, 63
- read_from, 63
- vertex, 64
- vertex_count, 64
- pos
 - GameAsteroids::Actor, 19
- print_to
 - Circle, 23
 - Polygon, 63
 - Shape, 73
- radius
 - Circle, 24
- randd
 - anonymous_namespace{game_asteroids.cpp}, 9
- randomize_spike_height
 - GameFlappyBird, 44
- read_from
 - Circle, 23
 - Polygon, 63
 - Shape, 73
- reference
 - DynArray< T >::Iterator, 56
- render
 - Screen, 67
- RIGHT
 - Vector, 86
- rot
 - GameAsteroids::Actor, 19
- rotate
 - Vector, 85
- rotate_around
 - Vector, 85
- save
 - gtest_lite::ostreamRedir, 59
- Screen, 64
 - ~Screen, 66
 - clear, 67
 - data, 68
 - draw_shape, 67
 - height, 68
 - idx, 67
 - operator=, 67
 - render, 67
 - Screen, 66
 - size, 68
 - width, 68
- screen.cpp, 113
- screen.h, 114
 - operator<<, 116
- seconds_since_asteroid
 - GameAsteroids, 41
- seconds_till_snake_jumps
 - GameSnake, 50
- seconds_until_asteroid
 - GameAsteroids, 41
- Segment, 69
 - a, 70
 - b, 71
 - closest_point_to, 70
 - is_point_to_the_left, 70
 - Segment, 70
- segment.cpp, 116
- segment.h, 116
 - operator<<, 117
- Shape, 71
 - ~Shape, 72
 - has_point, 72
 - intersects_with, 73
 - print_to, 73
 - read_from, 73
- shape.h, 118
 - operator<<, 119
 - operator>>, 119
- shape_circle.h, 119
- shape_parser
 - GameSnake, 50
- shape_parser.cpp, 121
- shape_parser.h, 121
- shape_polygon.cpp, 122
- shape_polygon.h, 123
- ShapeParser, 74
 - ~ShapeParser, 76
 - array, 76
 - begin, 76
 - end, 76
 - operator=, 76
 - ShapeParser, 75
 - SuperIt, 75
- ShapeParser::Iterator, 51
 - Iterator, 53
 - operator*, 53
 - value_type, 53
- shapes.h, 125
- size
 - DynArray< T >, 34
 - GameAsteroids::Actor, 19
 - Screen, 68
- SNAKE_LENGTH
 - GameSnake, 50
- snake_level.txt, 125
- snake_pos
 - GameSnake, 50
- spawn_random_asteroid
 - GameAsteroids, 39
- speed
 - GameAsteroids::Actor, 19
- spike_x
 - GameFlappyBird, 45
- spike_x_speed
 - GameFlappyBird, 46
- spike_y_offset
 - GameFlappyBird, 46
- src
 - gtest_lite::ostreamRedir, 59
- status

- gtest_lite::Test, [81](#)
- SUCCEED
 - gtest_lite.h, [107](#)
- sum
 - gtest_lite::Test, [81](#)
- SuperIt
 - ShapeParser, [75](#)
- TEST
 - gtest_lite.h, [107](#)
- Test
 - gtest_lite::Test, [78](#)
- test
 - gtest_lite, [14](#)
- tmp
 - gtest_lite::Test, [81](#)
- trCode
 - Console, [28](#)
- UP
 - Vector, [86](#)
- update
 - Game, [36](#)
 - GameAsteroids, [40](#)
 - GameAsteroids::Actor, [18](#)
 - GameFlappyBird, [44](#)
 - GameSnake, [49](#)
- upper_wall
 - GameFlappyBird, [46](#)
- value_type
 - DynArray< T >::Iterator, [56](#)
 - ShapeParser::Iterator, [53](#)
- Vector, [82](#)
 - angle_to, [83](#)
 - distance_squared_to, [84](#)
 - distance_to, [84](#)
 - DOWN, [86](#)
 - LEFT, [86](#)
 - operator*, [84](#)
 - operator+, [84](#)
 - operator+=, [84](#)
 - operator-, [85](#)
 - polar, [85](#)
 - RIGHT, [86](#)
 - rotate, [85](#)
 - rotate_around, [85](#)
 - UP, [86](#)
 - Vector, [83](#)
 - x, [86](#)
 - y, [86](#)
- vector.cpp, [125](#)
- vector.h, [126](#)
 - operator<<, [127](#)
 - operator>>, [127](#)
- vectormath.h, [127](#)
 - MATH_PI, [128](#)
- vertex
 - Polygon, [64](#)
- vertex_count
 - Polygon, [64](#)
- width
 - Screen, [68](#)
- x
 - Vector, [86](#)
- y
 - Vector, [86](#)