Flappy Bird Game

Gerado por Doxygen 1.9.1

1 Flappy Bird Game 1

2 Índice Hierárquico 3

2.1 Hierarquia de Classes . . . 3

3 Índice dos Componentes 5

3.1 Lista de Classes . . . 5

4 Índice dos Arquivos 7

4.1 Lista de Arquivos . . . 7

5 Classes 9

5.1 Referência da Classe AllegroController . . . 9

5.1.1 Descrição detalhada . . . 10

5.1.2 Construtores e Destrutores . . . 10

5.1.2.1 AllegroController() . . . 10

5.1.2.2 ~AllegroController() . . . 10

5.1.3 Funções membros . . . 10

5.1.3.1 cleanup() . . . 11

5.1.3.2 initialize() . . . 11

5.1.3.3 load() . . . 11

5.1.3.4 run() . . . 11

5.1.4 Atributos . . . 12

5.1.4.1 current_state . . . 12

5.1.4.2 game . . . 12

5.1.4.3 motion . . . 12

5.1.4.4 screen_height . . . 12

5.1.4.5 screen_width . . . 12

5.2 Referência da Classe Bird . . . 13

5.2.1 Descrição detalhada . . . 14

5.2.2 Construtores e Destrutores . . . 14

5.2.2.1 Bird() . . . 15

5.2.2.2 ~Bird() . . . 15

5.2.3 Funções membros . . . 15

5.2.3.1 check_bird_collision() . . . 15

5.2.3.2 destroy_bitmaps() . . . 16

5.2.3.3 draw() . . . 16

5.2.3.4 jump() . . . 17

5.2.3.5 loop_animation() . . . 17

5.2.3.6 reset_xy() . . . 18

5.2.3.7 set_break() . . . 18

5.2.3.8 set_by() . . . 19

5.2.3.9 update() . . . 20

5.2.4 Atributos . . . 20

5.2.4.1 breakanimation . . . 20
5.2.4.2 contB . . . 21
5.2.4.3 frame1 . . . 21
5.2.4.4 frame2 . . . 21
5.2.4.5 frame3 . . . 21
5.2.4.6 frames . . . 21
5.2.4.7 gravity . . . 21
5.2.4.8 jumpForce . . . 22
5.2.4.9 rotation . . . 22
5.2.4.10 valueB . . . 22
5.2.4.11 vy . . . 22
5.3 Referência da Estrutura Button . . . 22
5.3.1 Descrição detalhada . . . 23
5.3.2 Atributos . . . 23
5.3.2.1 buttonSelectState . . . 23
5.3.2.2 name . . . 23
5.4 Referência da Classe DifficultyMenu . . . 23
5.4.1 Descrição detalhada . . . 25
5.4.2 Construtores e Destrutores . . . 25
5.4.2.1 DifficultyMenu() . . . 25
5.4.3 Funções membros . . . 25
5.4.3.1 draw() . . . 25
5.4.3.2 enter() . . . 26
5.4.3.3 handle_input() . . . 26
5.4.3.4 update() . . . 26
5.4.4 Atributos . . . 27
5.4.4.1 buttonDifficultyEasy . . . 27
5.4.4.2 buttonDifficultyHard . . . 27
5.4.4.3 buttonDifficultyNormal . . . 27
5.4.4.4 buttonPositionSelected . . . 28
5.4.4.5 buttonSaveDeselect . . . 28
5.4.4.6 buttonSaveSelect . . . 28
5.4.4.7 difficultySelected . . . 28
5.4.4.8 font . . . 28
5.4.4.9 menuButtons . . . 28
5.5 Referência da Classe FlappyBird . . . 29
5.5.1 Descrição detalhada . . . 31
5.5.2 Construtores e Destrutores . . . 31
5.5.2.1 FlappyBird() . . . 31
5.5.3 Funções membros . . . 31
5.5.3.1 breaker() . . . 31
5.5.3.2 change_velocity() . . . 32

5.5.3.3 check_collisions() . . . 32
5.5.3.4 control_pipes() . . . 33
5.5.3.5 draw() . . . 33
5.5.3.6 draw_animated_ground() . . . 34
5.5.3.7 draw_HUD() . . . 34
5.5.3.8 draw_intial_text() . . . 35
5.5.3.9 get_bird() . . . 35
5.5.3.10 get_pipes() . . . 35
5.5.3.11 get_state() . . . 36
5.5.3.12 jump() . . . 36
5.5.3.13 reset() . . . 36
5.5.3.14 saveCurrentPlayerScore() . . . 37
5.5.3.15 set_current_player() . . . 37
5.5.3.16 set_playerscore() . . . 37
5.5.3.17 starter() . . . 38
5.5.3.18 unbreaker() . . . 38
5.5.3.19 update() . . . 38
5.5.3.20 update_score() . . . 39
5.5.4 Atributos . . . 40
5.5.4.1 bird1 . . . 40
5.5.4.2 bird2 . . . 40
5.5.4.3 bird3 . . . 40
5.5.4.4 change_vel . . . 40
5.5.4.5 currentPlayer . . . 41
5.5.4.6 difficulty_game . . . 41
5.5.4.7 flappy_obj . . . 41
5.5.4.8 gameOverSound . . . 41
5.5.4.9 ground . . . 41
5.5.4.10 ground2 . . . 41
5.5.4.11 pipe . . . 42
5.5.4.12 pipelist . . . 42
5.5.4.13 pointSound . . . 42
5.5.4.14 positionF2_x . . . 42
5.5.4.15 positionF_x . . . 42
5.5.4.16 score . . . 42
5.5.4.17 state . . . 42
5.5.4.18 time . . . 43
5.5.4.19 velocity . . . 43
5.5.4.20 velocity_backup . . . 43
5.6 Referência da Classe GameObject . . . 43
5.6.1 Descrição detalhada . . . 44
5.6.2 Construtores e Destrutores . . . 44

5.6.2.1 GameObject() . . . 45
5.6.2.2 ~GameObject() . . . 45
5.6.3 Funções membros . . . 45
5.6.3.1 check_bird_collision() . . . 45
5.6.3.2 check_collision_with_boundaries() . . . 46
5.6.3.3 draw() . . . 46
5.6.3.4 get_height() . . . 46
5.6.3.5 get_width() . . . 47
5.6.3.6 get_x() . . . 47
5.6.3.7 get_x_final() . . . 47
5.6.3.8 get_y() . . . 48
5.6.3.9 set_finals() . . . 48
5.6.3.10 set_x() . . . 48
5.6.3.11 set_y() . . . 49
5.6.3.12 update() . . . 49
5.6.4 Atributos . . . 49
5.6.4.1 by . . . 49
5.6.4.2 height . . . 49
5.6.4.3 obj_sprite . . . 49
5.6.4.4 width . . . 49
5.6.4.5 x . . . 50
5.6.4.6 x_final . . . 50
5.6.4.7 y . . . 50
5.6.4.8 y_final . . . 50
5.7 Referência da Classe Image . . . 50
5.7.1 Descrição detalhada . . . 51
5.7.2 Construtores e Destrutores . . . 51
5.7.2.1 Image() [1/2] . . . 51
5.7.2.2 Image() [2/2] . . . 51
5.7.2.3 ~Image() . . . 52
5.7.3 Funções membros . . . 52
5.7.3.1 Draw() [1/2] . . . 52
5.7.3.2 Draw() [2/2] . . . 52
5.7.3.3 getBitmap() . . . 52
5.7.4 Atributos . . . 52
5.7.4.1 image . . . 53
5.7.4.2 x . . . 53
5.7.4.3 y . . . 53
5.8 Referência da Classe LeaderboardMenu . . . 53
5.8.1 Descrição detalhada . . . 54
5.8.2 Construtores e Destrutores . . . 54
5.8.2.1 LeaderboardMenu() . . . 55

5.8.3 Funções membros . . . 55
5.8.3.1 draw() . . . 55
5.8.3.2 enter() . . . 55
5.8.3.3 handle_input() . . . 56
5.8.3.4 update() . . . 56
5.8.4 Atributos . . . 57
5.8.4.1 campLeaderboard . . . 57
5.8.4.2 font . . . 57
5.9 Referência da Classe LoadName . . . 58
5.9.1 Descrição detalhada . . . 59
5.9.2 Construtores e Destrutores . . . 59
5.9.2.1 LoadName() . . . 59
5.9.3 Funções membros . . . 59
5.9.3.1 draw() . . . 59
5.9.3.2 enter() . . . 60
5.9.3.3 handle_input() . . . 60
5.9.3.4 update() . . . 61
5.9.4 Atributos . . . 62
5.9.4.1 buttonBackDeselect . . . 62
5.9.4.2 buttonBackSelect . . . 62
5.9.4.3 buttonInsertDeselect . . . 62
5.9.4.4 buttonInsertSelect . . . 62
5.9.4.5 buttonPositionSelected . . . 62
5.9.4.6 errorFont . . . 62
5.9.4.7 errorSituation . . . 62
5.9.4.8 menuButtons . . . 63
5.9.4.9 nameCampDeselect . . . 63
5.9.4.10 nameCampSelect . . . 63
5.9.4.11 nameFont . . . 63
5.9.4.12 playerNameString . . . 63
5.10 Referência da Classe MainMenu . . . 64
5.10.1 Descrição detalhada . . . 65
5.10.2 Construtores e Destrutores . . . 65
5.10.2.1 MainMenu() . . . 65
5.10.3 Funções membros . . . 65
5.10.3.1 draw() . . . 65
5.10.3.2 enter() . . . 66
5.10.3.3 handle_input() . . . 66
5.10.3.4 update() . . . 67
5.10.4 Atributos . . . 67
5.10.4.1 buttonDifficultyDeselect . . . 67
5.10.4.2 buttonDifficultySelect . . . 67

5.10.4.3 buttonExitDeselect . . . 68
5.10.4.4 buttonExitSelect . . . 68
5.10.4.5 buttonLeaderboardDeselect . . . 68
5.10.4.6 buttonLeaderboardSelect . . . 68
5.10.4.7 buttonLoadGameDeselect . . . 68
5.10.4.8 buttonLoadGameSelect . . . 68
5.10.4.9 buttonNewGameDeselect . . . 68
5.10.4.10 buttonNewGameSelect . . . 68
5.10.4.11 buttonPositionSelected . . . 69
5.10.4.12 buttonSettingsDeselect . . . 69
5.10.4.13 buttonSettingsSelect . . . 69
5.10.4.14 logoNormal . . . 69
5.10.4.15 menuButtons . . . 69
5.11 Referência da Classe Motion . . . 69
5.11.1 Descrição detalhada . . . 71
5.11.2 Construtores e Destrutores . . . 71
5.11.2.1 Motion() . . . 71
5.11.3 Funções membros . . . 71
5.11.3.1 draw() . . . 71
5.11.3.2 setController() . . . 72
5.11.3.3 update() . . . 72
5.11.4 Atributos . . . 73
5.11.4.1 background . . . 73
5.11.4.2 background_rain . . . 73
5.11.4.3 background_snow . . . 73
5.11.4.4 bird1 . . . 73
5.11.4.5 bird2 . . . 73
5.11.4.6 bird3 . . . 73
5.11.4.7 clouds . . . 73
5.11.4.8 clouds2 . . . 74
5.11.4.9 cont . . . 74
5.11.4.10 contB . . . 74
5.11.4.11 contF . . . 74
5.11.4.12 controll . . . 74
5.11.4.13 controller . . . 74
5.11.4.14 drips1 . . . 74
5.11.4.15 drips2 . . . 74
5.11.4.16 flakesBig . . . 75
5.11.4.17 flakesLittle . . . 75
5.11.4.18 flakesLittle2 . . . 75
5.11.4.19 ground . . . 75
5.11.4.20 ground2 . . . 75

5.11.4.21 lights . . . 75
5.11.4.22 little . . . 75
5.11.4.23 little2 . . . 75
5.11.4.24 little3 . . . 76
5.11.4.25 positionB_x . . . 76
5.11.4.26 positionC_x . . . 76
5.11.4.27 positionCC_x . . . 76
5.11.4.28 positionF . . . 76
5.11.4.29 positionF2 . . . 76
5.11.4.30 positionF2_x . . . 76
5.11.4.31 positionF3 . . . 76
5.11.4.32 positionF_x . . . 77
5.11.4.33 positionL_x . . . 77
5.11.4.34 positionR1 . . . 77
5.11.4.35 positionR2 . . . 77
5.11.4.36 speed_cloud . . . 77
5.11.4.37 speed_floor . . . 77
5.11.4.38 speed_light . . . 77
5.11.4.39 speed_little . . . 77
5.11.4.40 speedFlakes . . . 78
5.11.4.41 speedFlakes2 . . . 78
5.11.4.42 speedRain . . . 78
5.11.4.43 thunder . . . 78
5.11.4.44 thunder1 . . . 78
5.11.4.45 value . . . 78
5.12 Referência da Classe Music . . . 78
5.12.1 Descrição detalhada . . . 79
5.12.2 Construtores e Destrutores . . . 79
5.12.2.1 Music() . . . 79
5.12.2.2 ∼Music() . . . 79
5.12.3 Funções membros . . . 80
5.12.3.1 playMusic() . . . 80
5.12.4 Atributos . . . 80
5.12.4.1 music . . . 80
5.13 Referência da Classe Pipe . . . 80
5.13.1 Descrição detalhada . . . 81
5.13.2 Construtores e Destrutores . . . 82
5.13.2.1 Pipe() . . . 82
5.13.3 Funções membros . . . 82
5.13.3.1 check_score() . . . 82
5.13.3.2 draw() . . . 83
5.13.3.3 get_y() . . . 83

5.13.3.4 is_off_screen() . . . 83
5.13.3.5 set_vx() . . . 83
5.13.3.6 set_y() . . . 84
5.13.3.7 update() . . . 84
5.13.4 Atributos . . . 84
5.13.4.1 scored . . . 84
5.13.4.2 vx . . . 84
5.14 Referência da Classe PipeList . . . 85
5.14.1 Descrição detalhada . . . 86
5.14.2 Construtores e Destrutores . . . 86
5.14.2.1 PipeList() . . . 86
5.14.3 Funções membros . . . 86
5.14.3.1 add_pipe_pair() . . . 86
5.14.3.2 check_collision() . . . 87
5.14.3.3 check_score() . . . 87
5.14.3.4 delete_pipe_pair() . . . 88
5.14.3.5 draw() . . . 88
5.14.3.6 get_pipe_pairs() . . . 89
5.14.3.7 get_points() . . . 89
5.14.3.8 reset() . . . 89
5.14.3.9 set_difficulty() . . . 90
5.14.3.10 set_start() . . . 90
5.14.3.11 set_vx() . . . 91
5.14.3.12 update() . . . 91
5.14.4 Atributos . . . 91
5.14.4.1 difficulty_pipe . . . 91
5.14.4.2 gen . . . 92
5.14.4.3 pipe1 . . . 92
5.14.4.4 Pipes . . . 92
5.14.4.5 points . . . 92
5.14.4.6 start . . . 92
5.15 Referência da Classe PipePair . . . 93
5.15.1 Descrição detalhada . . . 93
5.15.2 Construtores e Destrutores . . . 94
5.15.2.1 PipePair() [1/2] . . . 94
5.15.2.2 PipePair() [2/2] . . . 94
5.15.3 Atributos . . . 94
5.15.3.1 bottom . . . 94
5.15.3.2 movement . . . 95
5.15.3.3 signal . . . 95
5.15.3.4 top . . . 95
5.16 Referência da Classe Play . . . 95

5.16.1 Descrição detalhada . . . 96
5.16.2 Construtores e Destrutores . . . 97
5.16.2.1 Play() . . . 97
5.16.3 Funções membros . . . 97
5.16.3.1 draw() . . . 97
5.16.3.2 enter() . . . 97
5.16.3.3 handle_input() . . . 98
5.16.3.4 update() . . . 99
5.16.4 Atributos . . . 99
5.16.4.1 buttonExitDeselect . . . 99
5.16.4.2 buttonExitSelect . . . 99
5.16.4.3 buttonPause . . . 100
5.16.4.4 buttonPositionSelected . . . 100
5.16.4.5 buttonTryagainDeselect . . . 100
5.16.4.6 buttonTryagainSelect . . . 100
5.16.4.7 flappy . . . 100
5.16.4.8 font . . . 100
5.16.4.9 logoGameOver . . . 100
5.16.4.10 menuButtons . . . 101
5.16.4.11 status . . . 101
5.17 Referência da Classe Player . . . 101
5.17.1 Descrição detalhada . . . 102
5.17.2 Construtores e Destrutores . . . 102
5.17.2.1 Player() . . . 102
5.17.3 Funções membros . . . 102
5.17.3.1 CheckingName() . . . 102
5.17.3.2 GetName() . . . 103
5.17.3.3 GetScore() . . . 104
5.17.3.4 operator<() . . . 104
5.17.3.5 ReadLeaderboard() . . . 104
5.17.3.6 SaveLeaderboard() [1/2] . . . 105
5.17.3.7 SaveLeaderboard() [2/2] . . . 106
5.17.3.8 SetScore() . . . 106
5.17.3.9 ShowLeaderboard() . . . 107
5.17.3.10 SortLeaderboard() . . . 107
5.17.4 Atributos . . . 107
5.17.4.1 name . . . 107
5.17.4.2 score . . . 107
5.18 Referência da Classe SettingsMenu . . . 108
5.18.1 Descrição detalhada . . . 109
5.18.2 Construtores e Destrutores . . . 109
5.18.2.1 SettingsMenu() . . . 109

5.18.3 Funções membros . . . 109
5.18.3.1 draw() . . . 109
5.18.3.2 enter() . . . 110
5.18.3.3 handle_input() . . . 110
5.18.3.4 update() . . . 111
5.18.4 Atributos . . . 111
5.18.4.1 buttonBackDeselect . . . 111
5.18.4.2 buttonBackSelect . . . 112
5.18.4.3 buttonMusicDaySelect . . . 112
5.18.4.4 buttonMusicRainSelect . . . 112
5.18.4.5 buttonMusicSelect . . . 112
5.18.4.6 buttonMusicSnowSelect . . . 112
5.18.4.7 buttonNoMusicDaySelect . . . 112
5.18.4.8 buttonNoMusicRainSelect . . . 112
5.18.4.9 buttonNoMusicSnowSelect . . . 112
5.18.4.10 buttonPositionSelected . . . 113
5.18.4.11 campSettingsMusic . . . 113
5.18.4.12 campSettingsNoMusic . . . 113
5.18.4.13 font . . . 113
5.18.4.14 menuButtons . . . 113
5.18.4.15 musicState . . . 113
5.18.4.16 weatherSelected . . . 114
5.19 Referência da Classe Sound . . . 114
5.19.1 Descrição detalhada . . . 114
5.19.2 Construtores e Destrutores . . . 114
5.19.2.1 Sound() . . . 114
5.19.2.2 ∼Sound() . . . 115
5.19.3 Funções membros . . . 115
5.19.3.1 playSound() . . . 115
5.19.4 Atributos . . . 115
5.19.4.1 sound . . . 115
5.20 Referência da Classe State . . . 116
5.20.1 Descrição detalhada . . . 116
5.20.2 Construtores e Destrutores . . . 116
5.20.2.1 ∼State() . . . 116
5.20.3 Funções membros . . . 117
5.20.3.1 draw() . . . 117
5.20.3.2 enter() . . . 117
5.20.3.3 handle_input() . . . 117
5.20.3.4 setGlobals() . . . 117
5.20.3.5 update() . . . 118
5.20.4 Atributos . . . 118

5.20.4.1 display . . . 118
5.20.4.2 ev . . . 118
5.20.4.3 queue . . . 118
5.21 Referência da Classe TextFont . . . 118
5.21.1 Descrição detalhada . . . 119
5.21.2 Construtores e Destrutores . . . 119
5.21.2.1 TextFont() . . . 119
5.21.2.2 ∼TextFont() . . . 119
5.21.3 Funções membros . . . 119
5.21.3.1 setColor() . . . 120
5.21.3.2 writeText() . . . 120
5.21.4 Atributos . . . 120
5.21.4.1 b . . . 120
5.21.4.2 font . . . 120
5.21.4.3 g . . . 120
5.21.4.4 r . . . 121

6 Arquivos **123**

6.1 Referência do Arquivo docs/mainpage.dox . . . 123
6.2 Referência do Arquivo include/allegro_interface.hpp . . . 123
6.2.1 Descrição detalhada . . . 124
6.3 Referência do Arquivo include/assets.hpp . . . 124
6.3.1 Descrição detalhada . . . 125
6.3.2 Funções . . . 125
6.3.2.1 loadGlobalAssets() . . . 125
6.3.2.2 unloadGlobalAssets() . . . 126
6.3.3 Variáveis . . . 126
6.3.3.1 selectSound . . . 126
6.4 Referência do Arquivo include/bird.hpp . . . 126
6.4.1 Descrição detalhada . . . 127
6.5 Referência do Arquivo include/defines.hpp . . . 127
6.5.1 Descrição detalhada . . . 128
6.5.2 Definições e macros . . . 129
6.5.2.1 BIRD_VEL . . . 129
6.5.2.2 FPS . . . 129
6.5.2.3 GAP_SIZE . . . 129
6.5.2.4 GAP_X . . . 129
6.5.2.5 GRAVITY . . . 129
6.5.2.6 JUMP_FORCE . . . 129
6.5.2.7 MAX_INPUT_LENGTH . . . 129
6.5.2.8 OSCILATION . . . 130
6.5.2.9 PIPE_SPEED . . . 130

6.5.2.10 PIPE_VERTICAL_SPEED . . . 130
6.5.2.11 ROTATION . . . 130
6.5.2.12 SCREEN_H . . . 130
6.5.2.13 SCREEN_W . . . 130
6.5.2.14 TETO_BIRD . . . 130
6.5.2.15 TIME_GIF_BIRD . . . 130
6.5.2.16 X_INIT . . . 131
6.5.3 Enumerações . . . 131
6.5.3.1 ScreenState . . . 131
6.5.4 Variáveis . . . 131
6.5.4.1 MAX_ROTATION_DOWN . . . 131
6.5.4.2 MAX_ROTATION_UP . . . 131
6.6 Referência do Arquivo include/flappy_bird_controller.hpp . . . 131
6.6.1 Descrição detalhada . . . 132
6.7 Referência do Arquivo include/game_object.hpp . . . 132
6.7.1 Descrição detalhada . . . 133
6.8 Referência do Arquivo include/init.hpp . . . 133
6.8.1 Descrição detalhada . . . 134
6.8.2 Funções . . . 135
6.8.2.1 deinit() . . . 135
6.8.2.2 init() . . . 135
6.8.3 Variáveis . . . 136
6.8.3.1 backgroundMusic . . . 136
6.8.3.2 difficulty . . . 136
6.8.3.3 display . . . 136
6.8.3.4 event_queue . . . 136
6.8.3.5 font . . . 136
6.8.3.6 g_sound_on . . . 137
6.8.3.7 icon . . . 137
6.8.3.8 keystate . . . 137
6.8.3.9 player . . . 137
6.8.3.10 ranking . . . 137
6.8.3.11 selectSound . . . 137
6.8.3.12 timer_FPS . . . 137
6.9 Referência do Arquivo include/motion.hpp . . . 138
6.9.1 Descrição detalhada . . . 138
6.10 Referência do Arquivo include/pipe.hpp . . . 138
6.10.1 Descrição detalhada . . . 139
6.11 Referência do Arquivo include/player.hpp . . . 140
6.11.1 Descrição detalhada . . . 140
6.12 Referência do Arquivo include/state.hpp . . . 140
6.12.1 Descrição detalhada . . . 141

6.13 Referência do Arquivo include/states/difficulty_menu.hpp . . . 141
6.13.1 Descrição detalhada . . . 142
6.14 Referência do Arquivo include/states/leaderboard_menu.hpp . . . 142
6.14.1 Descrição detalhada . . . 143
6.15 Referência do Arquivo include/states/load_game.hpp . . . 143
6.15.1 Descrição detalhada . . . 144
6.15.2 Enumerações . . . 144
6.15.2.1 insertNameSituations . . . 144
6.16 Referência do Arquivo include/states/main_menu.hpp . . . 145
6.16.1 Descrição detalhada . . . 145
6.16.2 Variáveis . . . 145
6.16.2.1 inputNameScreen . . . 146
6.17 Referência do Arquivo include/states/play.hpp . . . 146
6.17.1 Descrição detalhada . . . 146
6.18 Referência do Arquivo include/states/settings_menu.hpp . . . 147
6.18.1 Descrição detalhada . . . 147
6.19 Referência do Arquivo README.md . . . 148
6.20 Referência do Arquivo src/allegro_interface.cpp . . . 148
6.20.1 Descrição detalhada . . . 148
6.21 Referência do Arquivo src/assets.cpp . . . 148
6.21.1 Descrição detalhada . . . 149
6.21.2 Funções . . . 149
6.21.2.1 loadGlobalAssets() . . . 149
6.21.2.2 unloadGlobalAssets() . . . 149
6.21.3 Variáveis . . . 150
6.21.3.1 selectSound . . . 150
6.22 Referência do Arquivo src/bird.cpp . . . 150
6.22.1 Descrição detalhada . . . 150
6.23 Referência do Arquivo src/flappy_bird_controller.cpp . . . 150
6.23.1 Descrição detalhada . . . 151
6.24 Referência do Arquivo src/game_object.cpp . . . 151
6.24.1 Descrição detalhada . . . 151
6.25 Referência do Arquivo src/init.cpp . . . 151
6.25.1 Descrição detalhada . . . 152
6.25.2 Definições e macros . . . 152
6.25.2.1 LOG . . . 152
6.25.3 Funções . . . 152
6.25.3.1 deinit() . . . 153
6.25.3.2 init() . . . 153
6.25.4 Variáveis . . . 154
6.25.4.1 backgroundMusic . . . 154
6.25.4.2 difficulty . . . 154

6.25.4.3 display . . . 154
6.25.4.4 event_queue . . . 154
6.25.4.5 font . . . 154
6.25.4.6 g_sound_on . . . 155
6.25.4.7 icon . . . 155
6.25.4.8 player . . . 155
6.25.4.9 ranking . . . 155
6.25.4.10 timer_FPS . . . 155
6.26 Referência do Arquivo src/main.cpp . . . 155
6.26.1 Descrição detalhada . . . 156
6.26.2 Funções . . . 156
6.26.2.1 main() . . . 156
6.27 Referência do Arquivo src/motion.cpp . . . 156
6.27.1 Descrição detalhada . . . 157
6.28 Referência do Arquivo src/pipe.cpp . . . 157
6.28.1 Descrição detalhada . . . 157
6.29 Referência do Arquivo src/player.cpp . . . 157
6.29.1 Descrição detalhada . . . 158
6.29.2 Funções . . . 158
6.29.2.1 operator<<() . . . 158
6.30 Referência do Arquivo src/state.cpp . . . 158
6.30.1 Descrição detalhada . . . 159
6.31 Referência do Arquivo src/states/difficulty_menu.cpp . . . 159
6.31.1 Descrição detalhada . . . 159
6.32 Referência do Arquivo src/states/leaderboard_menu.cpp . . . 159
6.32.1 Descrição detalhada . . . 160
6.33 Referência do Arquivo src/states/load_game.cpp . . . 160
6.33.1 Descrição detalhada . . . 160
6.34 Referência do Arquivo src/states/main_menu.cpp . . . 160
6.34.1 Descrição detalhada . . . 161
6.34.2 Variáveis . . . 161
6.34.2.1 inputNameScreen . . . 161
6.35 Referência do Arquivo src/states/play.cpp . . . 161
6.35.1 Descrição detalhada . . . 162
6.36 Referência do Arquivo src/states/settings_menu.cpp . . . 162
6.36.1 Descrição detalhada . . . 162

Índice Remissivo **163**

Capítulo 1

Flappy Bird Game

Trabalho acadêmico desenvolvido em C++17 com a biblioteca Allegro 5.

Funcionalidades principais:

- Estados de jogo (menu, dificuldade, ranking, jogo, pausa e game-over).
- Três temas climáticos: dia, neve e chuva (classe Motion).
- Persistência de ranking em arquivo de texto.
- Testes unitários com Doctest.
- Documentação automática via Doxygen.

Estrutura de diretórios:

- **src**/: código-fonte (.cpp)
- **include**/: headers (.hpp)
- **assets**/: sprites, fontes e sons
- **tests**/: casos de teste

Use o menu da esquerda para navegar por classes, arquivos e módulos.

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

AllegroController 9
Button 22
FlappyBird 29
GameObject 43
- Bird 13
- Pipe 80

Image 50
Motion 69
Music 78
PipeList 85
PipePair 93
Player 101
Sound 114
State 116
- DifficultyMenu 23
- LeaderboardMenu 53
- LoadName 58
- MainMenu 64
- Play 95
- SettingsMenu 108

TextFont 118

Capítulo 3

Índice dos Componentes

3.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

AllegroController
: Gerencia display, eventos e transições de estados do jogo 9

Bird
: Personagem controlável: gerencia movimento, animação de asas, pulo e detecção de colisões 13

Button
: Estrutura auxiliar que representa um botão de menu e seu estado de seleção 22

DifficultyMenu
: Gerencia a lógica e a renderização da tela de seleção de dificuldade 23

FlappyBird
: Controlador principal do jogo: orquestra objetos, HUD, pontuação e fluxo de estados 29

GameObject
: Objeto genérico do jogo contendo posição, sprite e hitbox 43

Image
: Encapsula carregamento e desenho de bitmaps 50

LeaderboardMenu
: Exibe ranking de jogadores e permite retorno ao menu principal 53

LoadName
: Tela de inserção de nome usada por Novo Jogo e Carregar Jogo 58

MainMenu
: Tela inicial do jogo, centraliza navegação para demais estados 64

Motion
: Responsável por atualizar e desenhar efeitos de parallax, clima e piso animado 69

Music
: Gerencia reprodução em loop de faixas de áudio 78

Pipe
: Obstáculo individual que se move horizontalmente e gera pontuação 80

PipeList
: Gerencia vetor de PipePair: spawn, atualização, colisão e score 85

PipePair
: Agrupa dois canos (superior e inferior) podendo ter movimento vertical 93

Play
: Estado principal de jogo: controla loop ativo, pontuação e game over 95

Player
: Armazena nome e pontuação e provê utilidades para ranking 101

SettingsMenu
: Tela de configurações: permite alterar clima e habilitar/desabilitar música 108

Sound
Carrega e reproduz samples de efeito sonoro 114
State
Define a interface que cada tela/estado deve implementar 116
TextFont
Wrapper para fontes bitmap/TTF e texto colorido 118

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

include/allegro_interface.hpp
: Declara o controlador que integra Allegro e a máquina de estados . . . 123

include/assets.hpp
: Declara classes utilitárias para carregar e usar imagens, sons e fontes no Allegro . . . 124

include/bird.hpp
: Declaracao da classe Bird (personagem controlavel) . . . 126

include/defines.hpp
: Constantes globais, macros e estruturas auxiliares do jogo . . . 127

include/flappy_bird_controller.hpp
: Declaracao da classe FlappyBird (controlador principal) . . . 131

include/game_object.hpp
: Classe base abstrata para todos os objetos renderizáveis . . . 132

include/init.hpp
: Declara funções de inicialização e finalização do Allegro . . . 133

include/motion.hpp
: Gerencia efeitos de parallax e animações de cenário . . . 138

include/pipe.hpp
: Declaration of pipe-related classes (Pipe, PipePair, PipeList) . . . 138

include/player.hpp
: Modela um jogador e operações de ranking . . . 140

include/state.hpp
: Interface base para estados da máquina de estados do jogo . . . 140

include/states/difficulty_menu.hpp
: Definição da classe DifficultyMenu, que representa o estado do menu de seleção de dificuldade 141

include/states/leaderboard_menu.hpp
: Definição da classe LeaderboardMenu, que representa o estado da tela de ranking . . . 142

include/states/load_game.hpp
: Definição da classe LoadName, que representa o estado de inserção de nome . . . 143

include/states/main_menu.hpp
: Definição da classe MainMenu, o estado que representa o menu principal do jogo . . . 145

include/states/play.hpp
: Declaracao da classe Play, responsavel pelo estado de jogo ativo . . . 146

include/states/settings_menu.hpp
: Definição da classe SettingsMenu, que representa o estado do menu de configurações . . . 147

src/allegro_interface.cpp
: Implementação do controlador principal que integra Allegro às lógicas de estado do jogo . . . 148

src/assets.cpp
Carregamento e gerenciamento de assets gráficos/áudio . . . 148
src/bird.cpp
Implementação da classe Bird (animação, física e colisões) . . . 150
src/flappy_bird_controller.cpp
Controlador de fluxo do jogo (pontuação, eventos globais) . . . 150
src/game_object.cpp
Métodos utilitários da classe base GameObject . . . 151
src/init.cpp
Inicialização de Allegro, fontes, addons e recursos globais . . . 151
src/main.cpp
Ponto de entrada do jogo Flappy Bird . . . 155
src/motion.cpp
Implementação da classe Motion (parallax e clima) . . . 156
src/pipe.cpp
Implementação das classes Pipe, PipePair e PipeList . . . 157
src/player.cpp
Implementação da classe Player e utilidades de ranking . . . 157
src/state.cpp
Implementação base de State para máquina de estados do jogo . . . 158
src/states/difficulty_menu.cpp
Implementação da classe DifficultyMenu, que gerencia a tela de seleção de dificuldade . . . 159
src/states/leaderboard_menu.cpp
Implementação da classe LeaderboardMenu, responsável por exibir a tela de ranking . . . 159
src/states/load_game.cpp
Implementação da classe LoadName, responsável pela tela de inserção de nome do jogador . 160
src/states/main_menu.cpp
Implementação da classe MainMenu, o estado principal e tela inicial do jogo . . . 160
src/states/play.cpp
Implementacao da classe Play . . . 161
src/states/settings_menu.cpp
Implementação da classe SettingsMenu, que gerencia a tela de configurações do jogo . . . 162

Capítulo 5

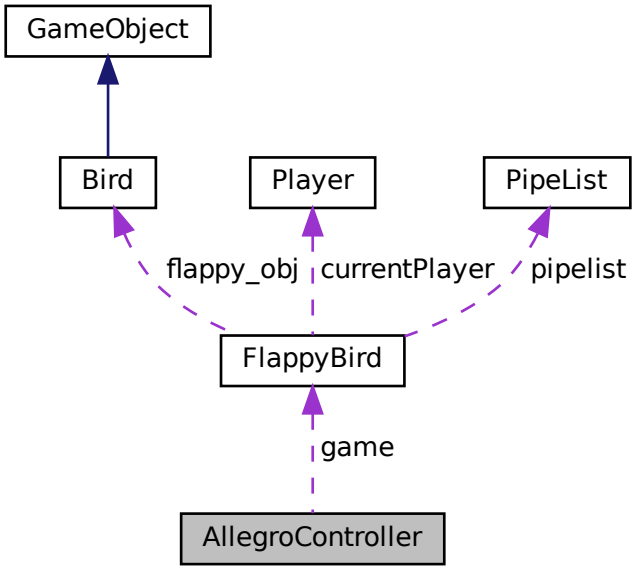
Classes

5.1 Referência da Classe AllegroController

Gerencia display, eventos e transições de estados do jogo.

`#include <allegro_interface.hpp>`

Diagrama de colaboração para AllegroController:



Membros Públicos

- AllegroController (float screen_w, float screen_h)
- ~AllegroController ()

  Destrói o controlador Allegro liberando recursos.
- bool initialize ()
- bool load ()
- void cleanup ()
- void run ()

  Executa o loop principal do jogo.

Atributos Privados

- int screen_width
- int screen_height
- std::unique_ptr< State > current_state
- std::unique_ptr< Motion > motion
- FlappyBird game

5.1.1 Descrição detalhada

Gerencia display, eventos e transições de estados do jogo.

5.1.2 Construtores e Destrutores

5.1.2.1 AllegroController()

```
AllegroController::AllegroController (
            float screen_w,
            float screen_h )
```

5.1.2.2 ~AllegroController()

```
AllegroController::~AllegroController ( )
```

Destrói o controlador Allegro liberando recursos.

5.1.3 Funções membros

5.1.3.1 cleanup()

```
void AllegroController::cleanup ( )
```

5.1.3.2 initialize()

```
bool AllegroController::initialize ( )
```

5.1.3.3 load()

```
bool AllegroController::load ( )
```

5.1.3.4 run()

```
void AllegroController::run ( )
```

Executa o loop principal do jogo.

O método é dividido em quatro fases recorrentes:

1. **Input** – aguarda um `ALLEGRO_EVENT` na fila e encaminha-o ao estado ativo para tratamento de entrada.
2. **Update** – em eventos de timer, avança a lógica do jogo e dos objetos via Motion.
3. **Render** – redesenha a cena quando necessário e a fila de eventos está vazia.
4. **State Transition** – se o estado ativo retornar um ponteiro diferente, realiza a troca, destruindo o estado anterior e chamando `enter()` no novo estado.

O loop continua até que `current_state` se torne `nullptr` (por exemplo, quando o usuário fecha a janela). Inicializa os componentes que o controller gerencia.

Chama o método de inicialização do primeiro estado.

Espera por um novo evento na fila. O programa fica bloqueado aqui até que algo aconteça, economizando CPU.

— Fase 1: Processar Entradas (Input) — Delega o tratamento do evento para o estado ativo atual.

— Fase 2: Atualizar a Lógica (Update) — A lógica do jogo avança em resposta a um evento do timer.

— Tratamento de Eventos Globais —

— Fase 3: Desenhar os Gráficos (Render) — Só redesenha se necessário e se não houver mais eventos na fila.

— Fase 4: Transição de Estado — Se handle_input ou update retornaram um novo estado, fazemos a transição.Esse é o diagrama das funções que utilizam essa função:

5.1.4 Atributos

5.1.4.1 current_state

```
std::unique_ptr<State> AllegroController::current_state  [private]
```

5.1.4.2 game

```
FlappyBird AllegroController::game  [private]
```

5.1.4.3 motion

```
std::unique_ptr<Motion> AllegroController::motion  [private]
```

5.1.4.4 screen_height

```
int AllegroController::screen_height  [private]
```

5.1.4.5 screen_width

```
int AllegroController::screen_width  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/allegro_interface.hpp
- src/allegro_interface.cpp

5.2 Referência da Classe Bird

Personagem controlável: gerencia movimento, animação de asas, pulo e detecção de colisões.

`#include <bird.hpp>`

Diagrama de hierarquia para Bird:

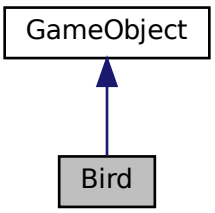
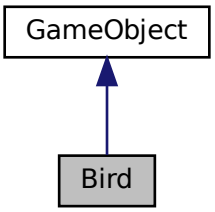


Diagrama de colaboração para Bird:



Membros Públicos

- Bird (ALLEGRO_BITMAP ∗img, float x, float y, ALLEGRO_BITMAP ∗img2, ALLEGRO_BITMAP ∗img3)
  Construtor.
- ∼Bird ()=default
  Destrutor padrao.
- void update ()
  Atualiza fisica e posicao.
- void draw () override
  Desenha sprite atual com rotacao.
- void jump ()
  Aplica impulso de pulo.
- void loop_animation (int type)

Gera animacao de bater de asas.

- bool check_bird_collision (const GameObject &other) const

  Verifica colisao com outro GameObject.
- void destroy_bitmaps ()

  Libera bitmap principal (quando necessario).
- void set_by ()

  Atualiza coordenada by (usada na HUD ou limites).
- void set_break (bool value)

  Habilita/desabilita animacao apos colisao.
- void reset_xy ()

  Reseta posicao inicial e velocidade.

Atributos Privados

- float gravity

  aceleracao gravitacional
- float jumpForce

  impulso vertical ao pular
- float vy = 0

  velocidade vertical
- float rotation = 0

  angulo de rotacao a desenhar
- float valueB = 0

  indice da animacao
- float contB = TIME_GIF_BIRD

  limite de frames por ciclo
- float frames = 0

  contador de frames
- ALLEGRO_BITMAP * frame1

  sprite asa para cima
- ALLEGRO_BITMAP * frame2

  sprite asa meio
- ALLEGRO_BITMAP * frame3

  sprite asa para baixo
- bool breakanimation = false

  pausa animacao (quando colide)

Outros membros herdados

5.2.1 Descrição detalhada

Personagem controlável: gerencia movimento, animação de asas, pulo e detecção de colisões.

5.2.2 Construtores e Destrutores

5.2.2.1 Bird()

```
Bird::Bird (
            ALLEGRO_BITMAP * img,
            float x,
            float y,
            ALLEGRO_BITMAP * img2,
            ALLEGRO_BITMAP * img3 )  [inline]
```

Construtor.

Parâmetros

img	primeiro frame
x	posicao inicial x
y	posicao inicial y
img2	segundo frame
img3	terceiro frame

5.2.2.2 ~Bird()

```
Bird::~Bird ( )  [default]
```

Destrutor padrao.

5.2.3 Funções membros

5.2.3.1 check_bird_collision()

```
bool Bird::check_bird_collision (
            const GameObject & other ) const  [virtual]
```

Verifica colisao com outro GameObject.

Parâmetros

other	objeto alvo

Retorna

true se colisao detectada

Verifica colisão com outro objeto considerando uma hitbox reduzida.

Parâmetros

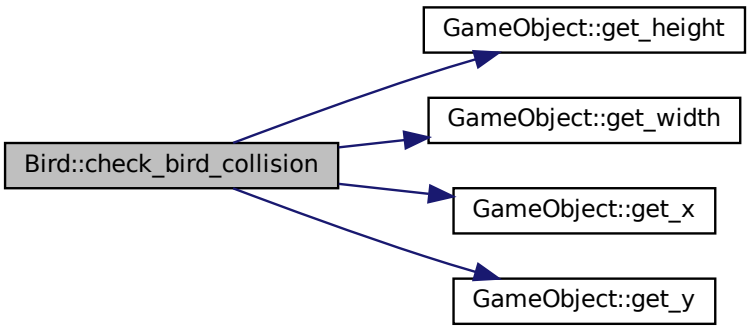
other	Objeto alvo.

Retorna

`true` se as hitboxes se sobrepõem.

Reimplementa GameObject.

Este é o diagrama das funções utilizadas por essa função:



5.2.3.2 destroy_bitmaps()

```
void Bird::destroy_bitmaps ( )
```

Libera bitmap principal (quando necessario).

Destrói o bitmap principal para liberar memória.

5.2.3.3 draw()

```
void Bird::draw ( )  [override], [virtual]
```

Desenha sprite atual com rotacao.

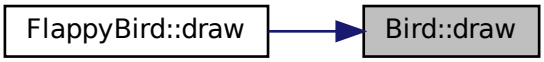
Desenha o pássaro com a animação ou rotação correspondente.

Seleciona entre a animação de entrada ou de voo conforme o estado do jogo e controla a rotação baseada em `vy`.

Implementa GameObject.

Este é o diagrama das funções utilizadas por essa função:

Esse é o diagrama das funções que utilizam essa função:

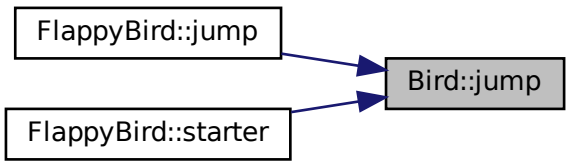


5.2.3.4 jump()

```
void Bird::jump ( )
```

Aplica impulso de pulo.

Aplica impulso vertical quando o jogador pressiona pular.Esse é o diagrama das funções que utilizam essa função:



5.2.3.5 loop_animation()

```
void Bird::loop_animation (
            int type )
```

Gera animacao de bater de asas.

Parâmetros

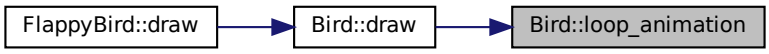
type	1 = intro; 2 = jogo ativo

Muda os frames de sprite para simular bater de asas.

Parâmetros

type	1 = animação de intro (pássaro voando da esquerda); 2 = animação durante o jogo.

Esse é o diagrama das funções que utilizam essa função:

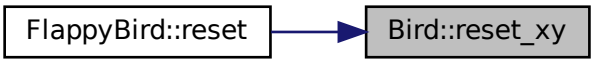


5.2.3.6 reset_xy()

```
void Bird::reset_xy ( )
```

Reseta posicao inicial e velocidade.

Reseta posição e velocidade do pássaro para valores iniciais.Esse é o diagrama das funções que utilizam essa função:



5.2.3.7 set_break()

```
void Bird::set_break (
            bool value )
```

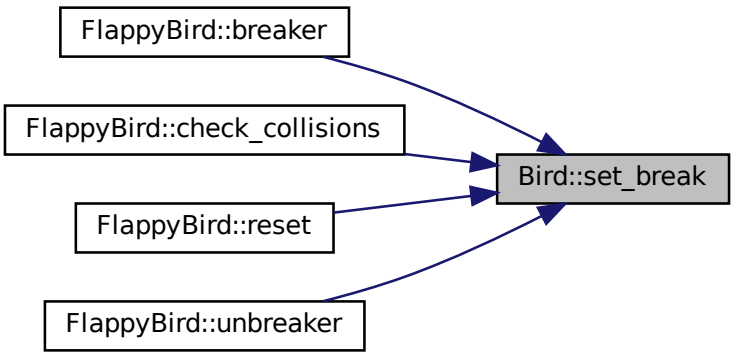
Habilita/desabilita animacao apos colisao.

Ativa/desativa a animação após colisão.

Parâmetros

value	`true` para pausar a animação.

Esse é o diagrama das funções que utilizam essa função:

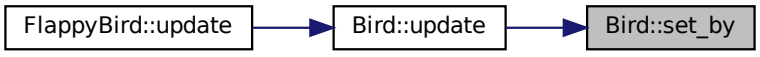


5.2.3.8 set_by()

```
void Bird::set_by ( )
```

Atualiza coordenada by (usada na HUD ou limites).

Atualiza a coordenada inferior `by` com base na altura.Esse é o diagrama das funções que utilizam essa função:



5.2.3.9 update()

```
void Bird::update ( )  [virtual]
```

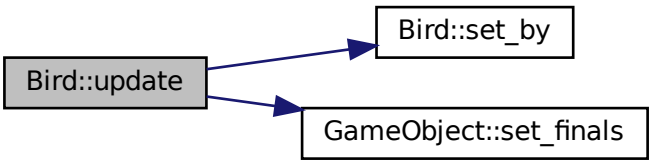
Atualiza fisica e posicao.

Atualiza física e posição do pássaro.

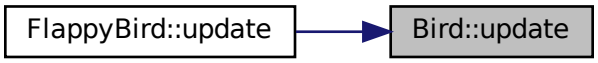
Move horizontalmente durante a animação de entrada até alcançar `X_INIT`. Depois aplica gravidade e velocidade vertical `vy`.

Implementa GameObject.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



5.2.4 Atributos

5.2.4.1 breakanimation

```
bool Bird::breakanimation = false  [private]
```

pausa animacao (quando colide)

5.2.4.2 contB

```
float Bird::contB = TIME_GIF_BIRD  [private]
```

limite de frames por ciclo

5.2.4.3 frame1

```
ALLEGRO_BITMAP* Bird::frame1  [private]
```

sprite asa para cima

5.2.4.4 frame2

```
ALLEGRO_BITMAP* Bird::frame2  [private]
```

sprite asa meio

5.2.4.5 frame3

```
ALLEGRO_BITMAP* Bird::frame3  [private]
```

sprite asa para baixo

5.2.4.6 frames

```
float Bird::frames = 0  [private]
```

contador de frames

5.2.4.7 gravity

```
float Bird::gravity  [private]
```

aceleracao gravitacional

5.2.4.8 jumpForce

```
float Bird::jumpForce  [private]
```

impulso vertical ao pular

5.2.4.9 rotation

```
float Bird::rotation = 0  [private]
```

angulo de rotacao a desenhar

5.2.4.10 valueB

```
float Bird::valueB = 0  [private]
```

indice da animacao

5.2.4.11 vy

```
float Bird::vy = 0  [private]
```

velocidade vertical

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/bird.hpp
- src/bird.cpp

5.3 Referência da Estrutura Button

Estrutura auxiliar que representa um botão de menu e seu estado de seleção.

```
#include <defines.hpp>
```

Atributos Públicos

- std::string name
- int buttonSelectState

5.3.1 Descrição detalhada

Estrutura auxiliar que representa um botão de menu e seu estado de seleção.

5.3.2 Atributos

5.3.2.1 buttonSelectState

```
int Button::buttonSelectState
```

5.3.2.2 name

```
std::string Button::name
```

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- include/defines.hpp

5.4 Referência da Classe DifficultyMenu

Gerencia a lógica e a renderização da tela de seleção de dificuldade.

```
#include <difficulty_menu.hpp>
```

Diagrama de hierarquia para DifficultyMenu:

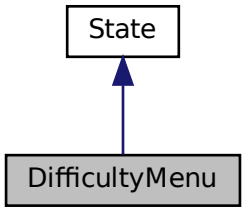
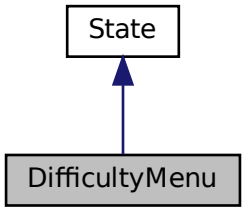


Diagrama de colaboração para DifficultyMenu:



Membros Públicos

- State ∗ handle_input (const ALLEGRO_EVENT &ev) override

  Processa a entrada do usuário (teclado).
- State ∗ update (Motion &motion) override

  Atualiza a lógica do estado a cada quadro.
- void draw (Motion &motion) override

  Desenha todos os elementos visuais do estado na tela.
- void enter () override

  Executa ações ao entrar neste estado do menu.
- DifficultyMenu ()

  Construtor da classe DifficultyMenu.

Atributos Públicos

- int buttonPositionSelected = 1
- std::string difficultySelected = "noOne"
- std::vector< Button > menuButtons

  Vetor que armazena os botões do menu e seus estados de seleção.

Atributos Privados

- std::unique_ptr< Image > buttonDifficultyEasy

  Ponteiros inteligentes para os recursos visuais (imagens e fontes) do menu.
- std::unique_ptr< Image > buttonDifficultyNormal
- std::unique_ptr< Image > buttonDifficultyHard
- std::unique_ptr< Image > buttonSaveSelect
- std::unique_ptr< Image > buttonSaveDeselect
- std::unique_ptr< TextFont > font

Outros membros herdados

5.4.1 Descrição detalhada

Gerencia a lógica e a renderização da tela de seleção de dificuldade.

Bibliotecas necessárias

Herda da classe base 'State' e implementa o comportamento específico para permitir que o usuário escolha e salve um nível de dificuldade para o jogo.

5.4.2 Construtores e Destrutores

5.4.2.1 DifficultyMenu()

```
DifficultyMenu::DifficultyMenu ( )
```

Construtor da classe DifficultyMenu.

Bibliotecas necessárias

Inicializa e carrega todos os recursos gráficos (imagens e fontes) necessários para a tela de seleção de dificuldade.

5.4.3 Funções membros

5.4.3.1 draw()

```
void DifficultyMenu::draw (
            Motion & motion )  [override], [virtual]
```

Desenha todos os elementos visuais do estado na tela.

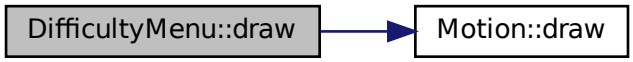
Parâmetros

motion	Referência ao objeto de controle para desenhar o fundo animado.

Renderiza a imagem de fundo e os botões de acordo com a seleção atual do usuário, além de um texto informativo.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.4.3.2 enter()

```
void DifficultyMenu::enter ( )  [override], [virtual]
```

Executa ações ao entrar neste estado do menu.

Garante que o menu de dificuldade seja inicializado em um estado padrão, com a dificuldade "Normal" pré-selecionada.

Implementa State.

5.4.3.3 handle_input()

```
State * DifficultyMenu::handle_input (
            const ALLEGRO_EVENT & ev )  [override], [virtual]
```

Processa a entrada do usuário (teclado).

Parâmetros

ev	O evento da Allegro a ser processado.

Retorna

Retorna um ponteiro para o próximo estado do jogo. Pode ser 'this' para continuar neste estado, 'nullptr' para fechar, ou um novo MainMenu para retornar.

Implementa State.

5.4.3.4 update()

```
State * DifficultyMenu::update (
            Motion & motion )  [override], [virtual]
```

Atualiza a lógica do estado a cada quadro.

Parâmetros

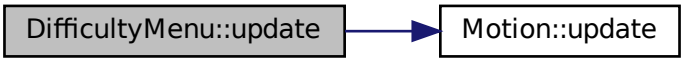
motion	Referência ao objeto de controle de animação do fundo.

Retorna

Retorna 'this' para indicar a permanência no estado atual.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.4.4 Atributos

5.4.4.1 buttonDifficultyEasy

```
std::unique_ptr<Image> DifficultyMenu::buttonDifficultyEasy  [private]
```

Ponteiros inteligentes para os recursos visuais (imagens e fontes) do menu.

5.4.4.2 buttonDifficultyHard

```
std::unique_ptr<Image> DifficultyMenu::buttonDifficultyHard  [private]
```

5.4.4.3 buttonDifficultyNormal

```
std::unique_ptr<Image> DifficultyMenu::buttonDifficultyNormal  [private]
```

5.4.4.4 buttonPositionSelected

```
int DifficultyMenu::buttonPositionSelected = 1
```

5.4.4.5 buttonSaveDeselect

```
std::unique_ptr<Image> DifficultyMenu::buttonSaveDeselect  [private]
```

5.4.4.6 buttonSaveSelect

```
std::unique_ptr<Image> DifficultyMenu::buttonSaveSelect  [private]
```

5.4.4.7 difficultySelected

```
std::string DifficultyMenu::difficultySelected = "noOne"
```

5.4.4.8 font

```
std::unique_ptr<TextFont> DifficultyMenu::font  [private]
```

5.4.4.9 menuButtons

```
std::vector<Button> DifficultyMenu::menuButtons
```

Valor inicial:

```
= {
      {"Easy", 0}, {"Normal", 1}, {"Hard", 0}, {"Save", 0}}
```

Vetor que armazena os botões do menu e seus estados de seleção.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

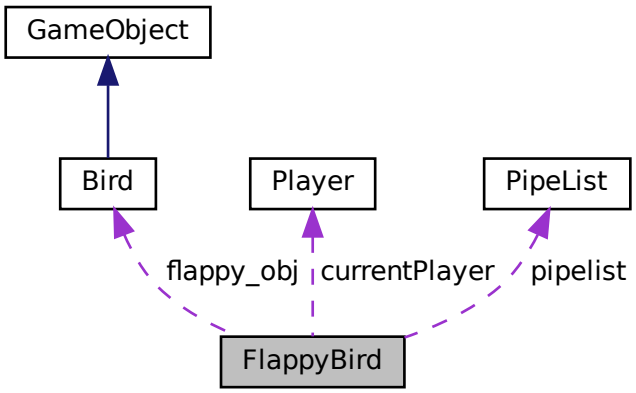
- include/states/difficulty_menu.hpp
- src/states/difficulty_menu.cpp

5.5 Referência da Classe FlappyBird

Controlador principal do jogo: orquestra objetos, HUD, pontuação e fluxo de estados.

`#include <flappy_bird_controller.hpp>`

Diagrama de colaboração para FlappyBird:



Membros Públicos

- FlappyBird ()
  Construtor padrao.
- void draw ()
  Desenha cena completa conforme estado.
- void update ()
  Atualiza logica principal (estado, objetos, pipes).
- void reset ()
  Reseta jogo para configuracao inicial.
- void set_current_player (Player &player)
  Define jogador atual (para leaderboard).
- void saveCurrentPlayerScore ()
  Salva pontuacao no leaderboard externo.
- bool check_collisions ()
  Verifica colisao entre passaro, limites ou canos.
- void jump ()
  Comanda pulo do passaro (estado 2).
- void starter ()
  Transita do estado 1 para 2 e aplica primeiro pulo.
- void control_pipes ()
  Adiciona / remove canos conforme necessidade.
- void draw_intial_text ()
  Desenha texto inicial "PRESS SPACE TO PLAY".

- void update_score ()
  Atualiza pontuacao quando passaro passa canos.
- void change_velocity ()
  Aumenta velocidade conforme pontuacao.
- void draw_HUD ()
  Desenha HUD de pontuacao.
- void draw_animated_ground (float velocity)
  Desenha solo em loop para criar efeito de movimento.
- void breaker ()
  Pausa fisica e anima passaro colidido.
- void unbreaker ()
  Retoma jogo apos pausa.
- int get_state ()
  Retorna estado atual (0,1,2).
- void set_playerscore ()
  Atualiza score no objeto Player.
- const Bird & get_bird () const
- const PipeList & get_pipes () const

Atributos Privados

- std::unique_ptr< Image > bird1
  sprite 1 do passaro
- std::unique_ptr< Image > bird2
  sprite 2 do passaro
- std::unique_ptr< Image > bird3
  sprite 3 do passaro
- std::unique_ptr< Image > pipe
  sprite do cano
- std::unique_ptr< Image > ground
  sprite do solo
- std::unique_ptr< Image > ground2
  sprite do solo (loop)
- std::unique_ptr< Sound > pointSound
  som de pontuacao
- std::unique_ptr< Sound > gameOverSound
  som de game over
- Bird flappy_obj
  instancia controlavel do passaro
- PipeList pipelist
  lista de canos ativos
- int state = 0
  0 = intro, 1 = aguardando, 2 = jogando
- float time = 0
- float velocity = PIPE_SPEED
- float velocity_backup = 0
- int score = 0
- float positionF_x = 0
- float positionF2_x = SCREEN_W
- int change_vel = 2
  frequencia de aumento de velocidade
- int difficulty_game = difficulty
- Player * currentPlayer = nullptr
  ponteiro para jogador atual

5.5.1 Descrição detalhada

Controlador principal do jogo: orquestra objetos, HUD, pontuação e fluxo de estados.

5.5.2 Construtores e Destrutores

5.5.2.1 FlappyBird()

```
FlappyBird::FlappyBird ( )
```

Construtor padrao.

Carrega assets, inicializa objetos e listas.

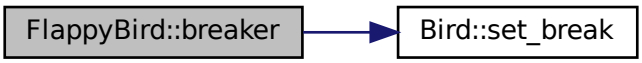
5.5.3 Funções membros

5.5.3.1 breaker()

```
void FlappyBird::breaker ( )
```

Pausa fisica e anima passaro colidido.

Pausa o movimento (usado em menus). Este é o diagrama das funções utilizadas por essa função:

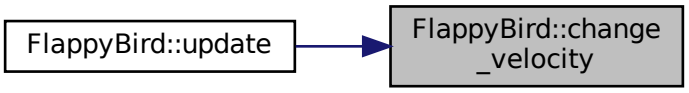


5.5.3.2 change_velocity()

```
void FlappyBird::change_velocity ( )
```

Aumenta velocidade conforme pontuacao.

Aumenta gradualmente a velocidade dos canos conforme pontuação. Esse é o diagrama das funções que utilizam essa função:



5.5.3.3 check_collisions()

```
bool FlappyBird::check_collisions ( )
```

Verifica colisao entre passaro, limites ou canos.

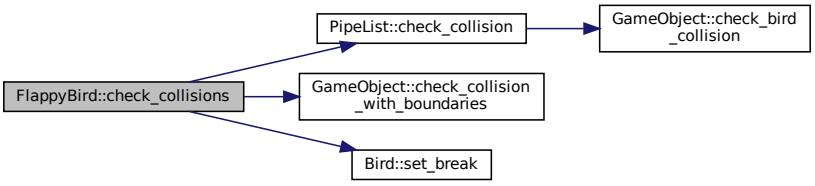
Verifica colisões do pássaro com canos ou limites da tela.

Retorna

true se colisao final detectada.

true se ocorrer colisão.

Este é o diagrama das funções utilizadas por essa função:

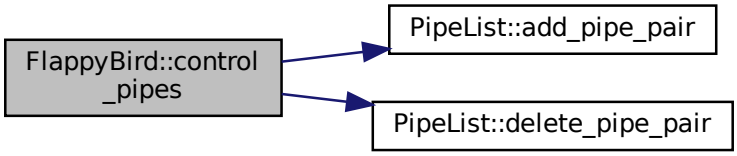


5.5.3.4 control_pipes()

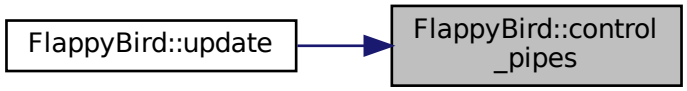
```
void FlappyBird::control_pipes ( )
```

Adiciona / remove canos conforme necessidade.

Gerencia criação e remoção de canos na tela. Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



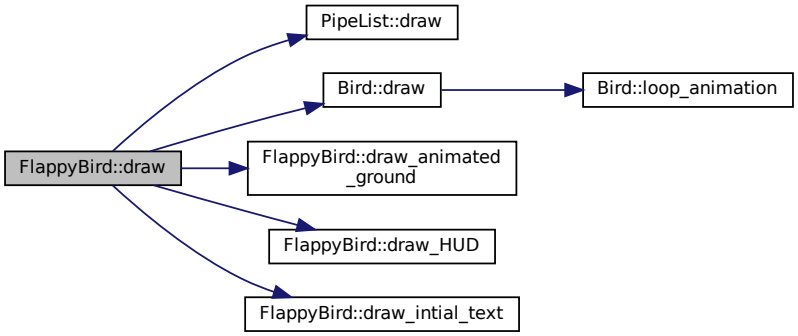
5.5.3.5 draw()

```
void FlappyBird::draw ( )
```

Desenha cena completa conforme estado.

Desenha todos os elementos do jogo de acordo com o estado atual. Este é o diagrama das funções utilizadas por

essa função:



5.5.3.6 draw_animated_ground()

```
void FlappyBird::draw_animated_ground (
            float velocity )
```

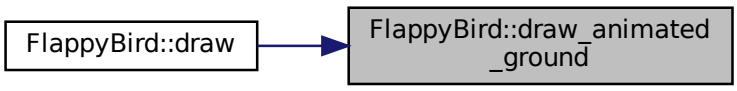
Desenha solo em loop para criar efeito de movimento.

Desenha o chão em rolagem contínua para efeito de movimento.

Parâmetros

velocity	velocidade horizontal a aplicar.
velocity	Velocidade de deslocamento.

Esse é o diagrama das funções que utilizam essa função:

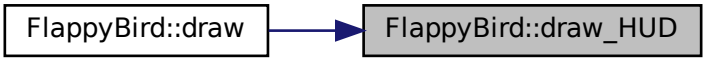


5.5.3.7 draw_HUD()

```
void FlappyBird::draw_HUD ( )
```

Desenha HUD de pontuacao.

Desenha a pontuação atual na tela. Esse é o diagrama das funções que utilizam essa função:



5.5.3.8 draw_intial_text()

```
void FlappyBird::draw_intial_text ( )
```

Desenha texto inicial "PRESS SPACE TO PLAY".

Desenha o texto piscante "PRESS SPACE TO PLAY". Esse é o diagrama das funções que utilizam essa função:

5.5.3.9 get_bird()

```
const Bird& FlappyBird::get_bird ( ) const  [inline]
```

5.5.3.10 get_pipes()

```
const PipeList& FlappyBird::get_pipes ( ) const  [inline]
```

5.5.3.11 get_state()

```
int FlappyBird::get_state ( )
```

Retorna estado atual (0,1,2).

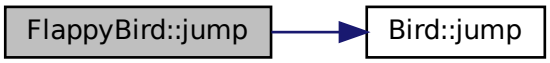
Retorna o estado interno atual.

5.5.3.12 jump()

```
void FlappyBird::jump ( )
```

Comanda pulo do passaro (estado 2).

Solicita pulo do pássaro se o jogo estiver em andamento. Este é o diagrama das funções utilizadas por essa função:

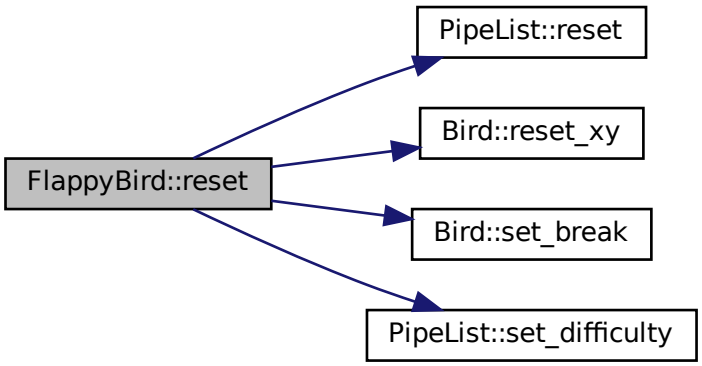


5.5.3.13 reset()

```
void FlappyBird::reset ( )
```

Reseta jogo para configuracao inicial.

Reinicia variáveis, pontos e objetos para começar uma nova partida. Este é o diagrama das funções utilizadas por essa função:

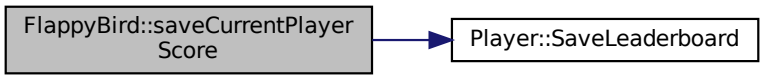


5.5.3.14 saveCurrentPlayerScore()

```
void FlappyBird::saveCurrentPlayerScore ( )
```

Salva pontuacao no leaderboard externo.

Salva o placar do jogador corrente no arquivo de ranking. Este é o diagrama das funções utilizadas por essa função:



5.5.3.15 set_current_player()

```
void FlappyBird::set_current_player (
            Player & player )
```

Define jogador atual (para leaderboard).

Define o jogador que receberá pontuação ao final da partida.

Parâmetros

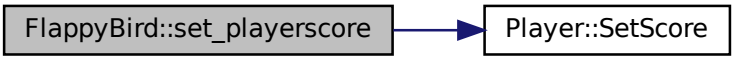
player	referencia ao objeto Player.

5.5.3.16 set_playerscore()

```
void FlappyBird::set_playerscore ( )
```

Atualiza score no objeto Player.

Copia a pontuação atual do jogo para o objeto Player. Este é o diagrama das funções utilizadas por essa função:

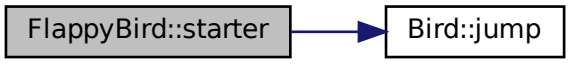


5.5.3.17 starter()

```
void FlappyBird::starter ( )
```

Transita do estado 1 para 2 e aplica primeiro pulo.

Inicia a partida a partir do estado de espera. Este é o diagrama das funções utilizadas por essa função:

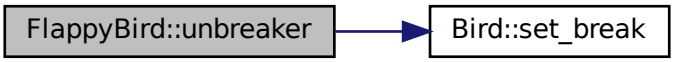


5.5.3.18 unbreaker()

```
void FlappyBird::unbreaker ( )
```

Retoma jogo apos pausa.

Restaura o movimento após pausa. Este é o diagrama das funções utilizadas por essa função:

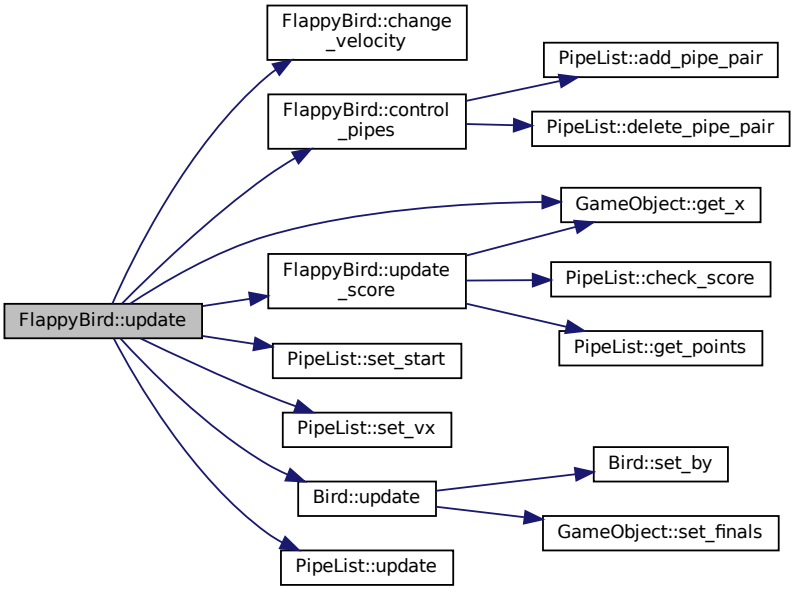


5.5.3.19 update()

```
void FlappyBird::update ( )
```

Atualiza logica principal (estado, objetos, pipes).

Atualiza lógica do jogo segundo o estado e aplica física. Este é o diagrama das funções utilizadas por essa função:

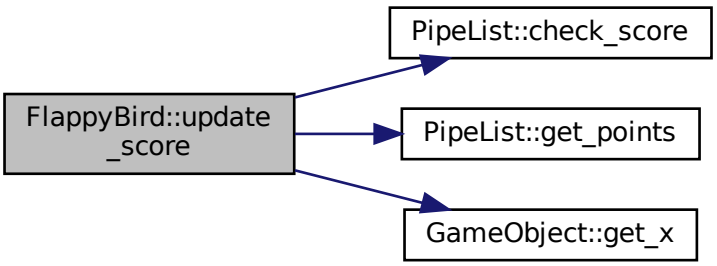


5.5.3.20 update_score()

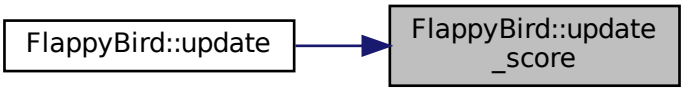
```
void FlappyBird::update_score ( )
```

Atualiza pontuacao quando passaro passa canos.

Atualiza contagem de pontos e toca som quando necessário. Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



5.5.4 Atributos

5.5.4.1 bird1

```
std::unique_ptr<Image> FlappyBird::bird1  [private]
```

sprite 1 do passaro

5.5.4.2 bird2

```
std::unique_ptr<Image> FlappyBird::bird2  [private]
```

sprite 2 do passaro

5.5.4.3 bird3

```
std::unique_ptr<Image> FlappyBird::bird3  [private]
```

sprite 3 do passaro

5.5.4.4 change_vel

```
int FlappyBird::change_vel = 2  [private]
```

frequencia de aumento de velocidade

5.5.4.5 currentPlayer

```
Player* FlappyBird::currentPlayer = nullptr  [private]
```

ponteiro para jogador atual

5.5.4.6 difficulty_game

```
int FlappyBird::difficulty_game = difficulty  [private]
```

5.5.4.7 flappy_obj

```
Bird FlappyBird::flappy_obj  [private]
```

instancia controlavel do passaro

5.5.4.8 gameOverSound

```
std::unique_ptr<Sound> FlappyBird::gameOverSound  [private]
```

som de game over

5.5.4.9 ground

```
std::unique_ptr<Image> FlappyBird::ground  [private]
```

sprite do solo

5.5.4.10 ground2

```
std::unique_ptr<Image> FlappyBird::ground2  [private]
```

sprite do solo (loop)

5.5.4.11 pipe

```
std::unique_ptr<Image> FlappyBird::pipe  [private]
```

sprite do cano

5.5.4.12 pipelist

```
PipeList FlappyBird::pipelist  [private]
```

lista de canos ativos

5.5.4.13 pointSound

```
std::unique_ptr<Sound> FlappyBird::pointSound  [private]
```

som de pontuacao

5.5.4.14 positionF2_x

```
float FlappyBird::positionF2_x = SCREEN_W  [private]
```

5.5.4.15 positionF_x

```
float FlappyBird::positionF_x = 0  [private]
```

5.5.4.16 score

```
int FlappyBird::score = 0  [private]
```

5.5.4.17 state

```
int FlappyBird::state = 0  [private]
```

0 = intro, 1 = aguardando, 2 = jogando

5.5.4.18 time

```
float FlappyBird::time = 0  [private]
```

5.5.4.19 velocity

```
float FlappyBird::velocity = PIPE_SPEED  [private]
```

5.5.4.20 velocity_backup

```
float FlappyBird::velocity_backup = 0  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

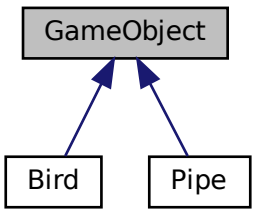
- include/flappy_bird_controller.hpp
- src/flappy_bird_controller.cpp

5.6 Referência da Classe GameObject

Objeto genérico do jogo contendo posição, sprite e hitbox.

```
#include <game_object.hpp>
```

Diagrama de hierarquia para GameObject:



Membros Públicos

- GameObject (ALLEGRO_BITMAP *img, float x, float y, float width, float height)
- virtual void update ()=0
- virtual void draw ()=0
- virtual bool check_bird_collision (const GameObject &other) const

  Verifica colisão entre este objeto e outro.
- bool check_collision_with_boundaries () const

  Detecta colisão do objeto com o teto ou chão da tela.
- void set_x (float new_x)

  Define coordenada X.
- void set_y (float new_y)

  Define coordenada Y.
- float get_x () const

  Obtém coordenada X.
- float get_y () const

  Obtém coordenada Y.
- float get_width () const
- float get_height () const
- void set_finals ()

  Atualiza limites inferiores (x_final, y_final) para colisão.
- float get_x_final () const

  Retorna coordenada X final (direita).
- virtual ~GameObject ()

Atributos Protegidos

- ALLEGRO_BITMAP * obj_sprite
- float x
- float y
- float by
- float width
- float height
- float y_final
- float x_final

5.6.1 Descrição detalhada

Objeto genérico do jogo contendo posição, sprite e hitbox.

Fornece utilidades de colisão com retângulos e getters padronizados. As coordenadas `(x, y)` representam o canto superior-esquerdo do sprite.

5.6.2 Construtores e Destrutores

5.6.2.1 GameObject()

```
GameObject::GameObject (
            ALLEGRO_BITMAP * img,
            float x,
            float y,
            float width,
            float height )  [inline]
```

5.6.2.2 ~GameObject()

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

5.6.3 Funções membros

5.6.3.1 check_bird_collision()

```
bool GameObject::check_bird_collision (
            const GameObject & other ) const  [virtual]
```

Verifica colisão entre este objeto e outro.

Parâmetros

other	Objeto alvo para teste de colisão.

Retorna

true se as áreas retangulares se sobrepõem.

Reimplementado por Bird.

Esse é o diagrama das funções que utilizam essa função:

5.6.3.2 check_collision_with_boundaries()

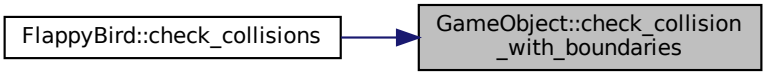
```
bool GameObject::check_collision_with_boundaries ( ) const
```

Detecta colisão do objeto com o teto ou chão da tela.

Retorna

true se colidir.

Esse é o diagrama das funções que utilizam essa função:



5.6.3.3 draw()

```
virtual void GameObject::draw ( )  [pure virtual]
```

Implementado por Pipe e Bird.

5.6.3.4 get_height()

```
float GameObject::get_height ( ) const  [inline]
```

Esse é o diagrama das funções que utilizam essa função:

5.6.3.5 get_width()

```
float GameObject::get_width ( ) const  [inline]
```

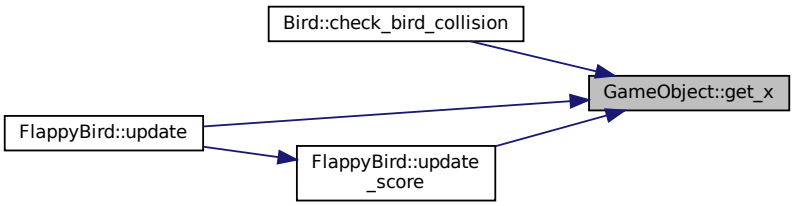
Esse é o diagrama das funções que utilizam essa função:

5.6.3.6 get_x()

```
float GameObject::get_x ( ) const
```

Obtém coordenada X.

Esse é o diagrama das funções que utilizam essa função:



5.6.3.7 get_x_final()

```
float GameObject::get_x_final ( ) const
```

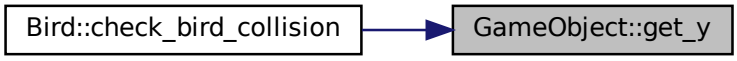
Retorna coordenada X final (direita).

5.6.3.8 get_y()

```
float GameObject::get_y ( ) const
```

Obtém coordenada Y.

Esse é o diagrama das funções que utilizam essa função:

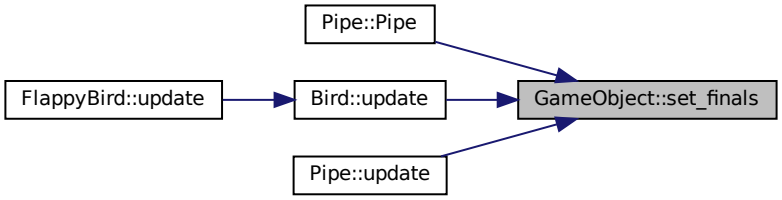


5.6.3.9 set_finals()

```
void GameObject::set_finals ( )
```

Atualiza limites inferiores (x_final, y_final) para colisão.

Esse é o diagrama das funções que utilizam essa função:



5.6.3.10 set_x()

```
void GameObject::set_x (
            float new_x )
```

Define coordenada X.

5.6.3.11 set_y()

```
void GameObject::set_y (
            float new_y )
```

Define coordenada Y.

5.6.3.12 update()

```
virtual void GameObject::update ( )  [pure virtual]
```

Implementado por Pipe e Bird.

5.6.4 Atributos

5.6.4.1 by

```
float GameObject::by  [protected]
```

5.6.4.2 height

```
float GameObject::height  [protected]
```

5.6.4.3 obj_sprite

```
ALLEGRO_BITMAP* GameObject::obj_sprite  [protected]
```

5.6.4.4 width

```
float GameObject::width  [protected]
```

5.6.4.5 x

```
float GameObject::x  [protected]
```

5.6.4.6 x_final

```
float GameObject::x_final  [protected]
```

5.6.4.7 y

```
float GameObject::y  [protected]
```

5.6.4.8 y_final

```
float GameObject::y_final  [protected]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/game_object.hpp
- src/game_object.cpp

5.7 Referência da Classe Image

Encapsula carregamento e desenho de bitmaps.

```
#include <assets.hpp>
```

Membros Públicos

- Image (const char *diretorio)

  Constrói uma imagem fora da tela.
- Image (const char *diretorio, float x, float y)

  Constrói uma imagem posicionada em coordenadas específicas.
- void Draw (float x, float y)

  Desenha a imagem nas coordenadas especificadas.
- void Draw ()

  Desenha a imagem usando as coordenadas internas armazenadas.
- ~Image ()

  Libera o bitmap associado.
- ALLEGRO_BITMAP * getBitmap ()

Atributos Privados

- ALLEGRO_BITMAP * image = NULL
- float x
- float y

5.7.1 Descrição detalhada

Encapsula carregamento e desenho de bitmaps.

5.7.2 Construtores e Destrutores

5.7.2.1 Image() [1/2]

```
Image::Image (
            const char * diretorio )
```

Constrói uma imagem fora da tela.

Parâmetros

diretorio	Caminho do arquivo de imagem.

Define a posição inicial fora da área visível para evitar flickering até que o programador defina coordenadas de desenho.

Exceções

std::bad_alloc	Se `diretorio` não puder ser carregado.

5.7.2.2 Image() [2/2]

```
Image::Image (
            const char * diretorio,
            float x,
            float y )
```

Constrói uma imagem posicionada em coordenadas específicas.

Parâmetros

diretorio	Caminho do bitmap.
x	Coordenada X.
y	Coordenada Y.

Exceções

std::bad_alloc	Se a imagem não puder ser carregada.

5.7.2.3 ∼Image()

```
Image::∼Image ( )
```

Libera o bitmap associado.

5.7.3 Funções membros

5.7.3.1 Draw() [1/2]

```
void Image::Draw ( )
```

Desenha a imagem usando as coordenadas internas armazenadas.

5.7.3.2 Draw() [2/2]

```
void Image::Draw (
            float x,
            float y )
```

Desenha a imagem nas coordenadas especificadas.

Parâmetros

x	Posição X.
y	Posição Y.

5.7.3.3 getBitmap()

```
ALLEGRO_BITMAP* Image::getBitmap ( )  [inline]
```

5.7.4 Atributos

5.7.4.1 image

```
ALLEGRO_BITMAP* Image::image = NULL  [private]
```

5.7.4.2 x

```
float Image::x  [private]
```

5.7.4.3 y

```
float Image::y  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/assets.hpp
- src/assets.cpp

5.8 Referência da Classe LeaderboardMenu

Exibe ranking de jogadores e permite retorno ao menu principal.

```
#include <leaderboard_menu.hpp>
```

Diagrama de hierarquia para LeaderboardMenu:

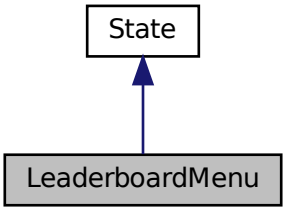
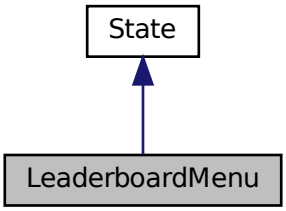


Diagrama de colaboração para LeaderboardMenu:



Membros Públicos

- State ∗ handle_input (const ALLEGRO_EVENT &ev) override

  Gerencia a entrada do usuário (teclado).
- State ∗ update (Motion &motion) override

  Atualiza a lógica do estado do movimento/animação do fundo.
- void draw (Motion &motion) override

  Desenha todos os elementos visuais do estado na tela.
- void enter () override

  Executado uma vez ao entrar no estado do Leaderboard.
- LeaderboardMenu ()

  Construtor da classe LeaderboardMenu.

Atributos Privados

- std::unique_ptr< TextFont > font

  Ponteiros inteligentes para os recursos visuais (imagens e fontes) do menu.
- std::unique_ptr< Image > campLeaderboard

Outros membros herdados

5.8.1 Descrição detalhada

Exibe ranking de jogadores e permite retorno ao menu principal.

Bibliotecas necessárias

5.8.2 Construtores e Destrutores

5.8.2.1 LeaderboardMenu()

```
LeaderboardMenu::LeaderboardMenu ( )
```

Construtor da classe LeaderboardMenu.

Bibliotecas necessárias

Carrega os recursos visuais necessários para a tela de ranking, como a imagem de fundo e a fonte do texto.

5.8.3 Funções membros

5.8.3.1 draw()

```
void LeaderboardMenu::draw (
            Motion & motion )  [override], [virtual]
```

Desenha todos os elementos visuais do estado na tela.

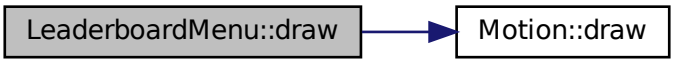
Parâmetros

motion	Referência ao objeto de controle para desenhar o fundo animado.

Renderiza a imagem de fundo e, em seguida, itera sobre o ranking para desenhar o nome e a pontuação de cada jogador em suas respectivas posições.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.8.3.2 enter()

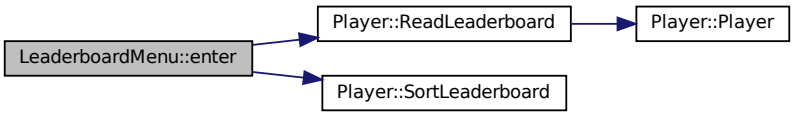
```
void LeaderboardMenu::enter ( )  [override], [virtual]
```

Executado uma vez ao entrar no estado do Leaderboard.

Carrega os dados do placar a partir de um arquivo e os ordena para exibição. Isso garante que a leitura e ordenação ocorram apenas uma vez por visualização.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.8.3.3 handle_input()

```
State * LeaderboardMenu::handle_input (
            const ALLEGRO_EVENT & ev )  [override], [virtual]
```

Gerencia a entrada do usuário (teclado).

Parâmetros

ev	O evento da Allegro a ser processado.

Retorna

Retorna um ponteiro para o próximo estado. Retorna 'this' para continuar no estado atual, 'nullptr' para fechar, ou um novo estado para transição.

Implementa State.

5.8.3.4 update()

```
State * LeaderboardMenu::update (
            Motion & motion )  [override], [virtual]
```

Atualiza a lógica do estado do movimento/animação do fundo.

Parâmetros

motion	Referência ao objeto de controle de animação do fundo.

Retorna

Retorna 'this' para indicar a permanência no estado atual.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:

5.8.4 Atributos

5.8.4.1 campLeaderboard

```
std::unique_ptr<Image> LeaderboardMenu::campLeaderboard  [private]
```

5.8.4.2 font

```
std::unique_ptr<TextFont> LeaderboardMenu::font  [private]
```

Ponteiros inteligentes para os recursos visuais (imagens e fontes) do menu.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/states/leaderboard_menu.hpp
- src/states/leaderboard_menu.cpp

5.9 Referência da Classe LoadName

Tela de inserção de nome usada por Novo Jogo e Carregar Jogo.

`#include <load_game.hpp>`

Diagrama de hierarquia para LoadName:

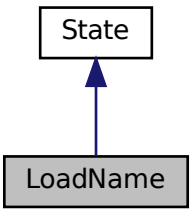
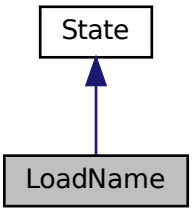


Diagrama de colaboração para LoadName:



Membros Públicos

- State ∗ handle_input (const ALLEGRO_EVENT &ev) override

  Gerencia toda a entrada de teclado do usuário.
- State ∗ update (Motion &motion) override

  Atualiza a lógica do estado do movimento/animação de fundo a cada quadro.
- void draw (Motion &motion) override

  Desenha todos os elementos visuais do estado na tela.
- void enter () override

  Executado ao entrar no estado. Prepara a tela para uma nova interação.
- LoadName ()

  Construtor da classe LoadName.

Atributos Públicos

- std::string playerNameString = ""
- int buttonPositionSelected = 2
- int errorSituation = noError
- std::vector< Button > menuButtons

  Vetor que armazena os botões do menu e seus estados de seleção.

Atributos Privados

- std::unique_ptr< Image > buttonBackSelect

  Ponteiros inteligentes para os recursos visuais (imagens e fontes) da tela.
- std::unique_ptr< Image > buttonBackDeselect
- std::unique_ptr< Image > buttonInsertSelect
- std::unique_ptr< Image > buttonInsertDeselect
- std::unique_ptr< Image > nameCampSelect
- std::unique_ptr< Image > nameCampDeselect
- std::unique_ptr< TextFont > nameFont
- std::unique_ptr< TextFont > errorFont

Outros membros herdados

5.9.1 Descrição detalhada

Tela de inserção de nome usada por Novo Jogo e Carregar Jogo.

5.9.2 Construtores e Destrutores

5.9.2.1 LoadName()

```
LoadName::LoadName ( )
```

Construtor da classe LoadName.

Bibliotecas necessárias

Carrega todos os recursos gráficos (imagens, fontes) e sonoros necessários para a tela de inserção de nome.

5.9.3 Funções membros

5.9.3.1 draw()

```
void LoadName::draw (
            Motion & motion )  [override], [virtual]
```

Desenha todos os elementos visuais do estado na tela.

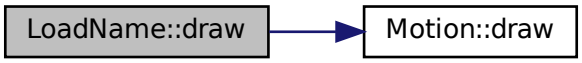
Parâmetros

motion	Referência para desenhar o fundo animado.

Renderiza os botões, o campo de texto com o nome do jogador e as mensagens de erro, se houver, com base no estado atual da interação.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.9.3.2 enter()

```
void LoadName::enter ( )  [override], [virtual]
```

Executado ao entrar no estado. Prepara a tela para uma nova interação.

Limpa o nome do jogador e qualquer mensagem de erro de uma visita anterior e redefine o foco do menu para o campo de inserção de nome.

Implementa State.

5.9.3.3 handle_input()

```
State * LoadName::handle_input (
            const ALLEGRO_EVENT & ev )  [override], [virtual]
```

Gerencia toda a entrada de teclado do usuário.

Parâmetros

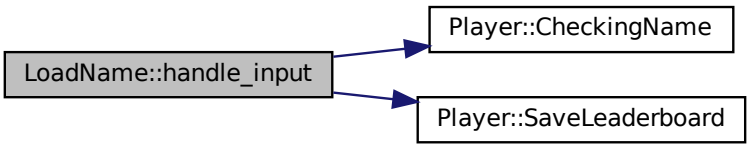
ev	O evento da Allegro a ser processado.

Retorna

State* Ponteiro para o próximo estado. Retorna 'nullptr' para fechar, 'new MainMenu()' ou 'new Play()' para transição, ou 'this' para permanecer.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.9.3.4 update()

```
State * LoadName::update (
            Motion & motion )  [override], [virtual]
```

Atualiza a lógica do estado do movimento/animação de fundo a cada quadro.

Parâmetros

motion	Referência ao objeto de controle de animação do fundo.

Retorna

State* Retorna 'this' para indicar a permanência no estado atual.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:

5.9.4 Atributos

5.9.4.1 buttonBackDeselect

```
std::unique_ptr<Image> LoadName::buttonBackDeselect  [private]
```

5.9.4.2 buttonBackSelect

```
std::unique_ptr<Image> LoadName::buttonBackSelect  [private]
```

Ponteiros inteligentes para os recursos visuais (imagens e fontes) da tela.

5.9.4.3 buttonInsertDeselect

```
std::unique_ptr<Image> LoadName::buttonInsertDeselect  [private]
```

5.9.4.4 buttonInsertSelect

```
std::unique_ptr<Image> LoadName::buttonInsertSelect  [private]
```

5.9.4.5 buttonPositionSelected

```
int LoadName::buttonPositionSelected = 2
```

5.9.4.6 errorFont

```
std::unique_ptr<TextFont> LoadName::errorFont  [private]
```

5.9.4.7 errorSituation

```
int LoadName::errorSituation = noError
```

5.9.4.8 menuButtons

```
std::vector<Button> LoadName::menuButtons
```

Valor inicial:

```
= {
      {"Back", 0}, {"Insert", 0}, {"NameCamp", 1}}
```

Vetor que armazena os botões do menu e seus estados de seleção.

5.9.4.9 nameCampDeselect

```
std::unique_ptr<Image> LoadName::nameCampDeselect  [private]
```

5.9.4.10 nameCampSelect

```
std::unique_ptr<Image> LoadName::nameCampSelect  [private]
```

5.9.4.11 nameFont

```
std::unique_ptr<TextFont> LoadName::nameFont  [private]
```

5.9.4.12 playerNameString

```
std::string LoadName::playerNameString = ""
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/states/load_game.hpp
- src/states/load_game.cpp

5.10 Referência da Classe MainMenu

Tela inicial do jogo, centraliza navegação para demais estados.

`#include <main_menu.hpp>`

Diagrama de hierarquia para MainMenu:

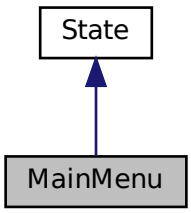
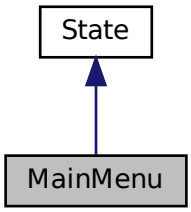


Diagrama de colaboração para MainMenu:



Membros Públicos

- State * handle_input (const ALLEGRO_EVENT &ev) override

  Gerencia a entrada do teclado para navegar e selecionar opções no menu.
- State * update (Motion &motion) override

  Atualiza a lógica do estado do movimento/animação de fundo a cada quadro.
- void draw (Motion &motion) override

  Desenha todos os elementos visuais do menu principal na tela.
- void enter () override

  Prepara o menu para exibição.
- MainMenu ()

  Construtor da classe MainMenu.

Atributos Públicos

- int buttonPositionSelected = 0
- std::vector< Button > menuButtons

  Vetor que armazena os botões do menu e seus estados de seleção.

Atributos Privados

- std::unique_ptr< Image > logoNormal

  Ponteiros inteligentes para todos os recursos visuais (imagens) do menu.
- std::unique_ptr< Image > buttonNewGameSelect
- std::unique_ptr< Image > buttonNewGameDeselect
- std::unique_ptr< Image > buttonLoadGameSelect
- std::unique_ptr< Image > buttonLoadGameDeselect
- std::unique_ptr< Image > buttonSettingsSelect
- std::unique_ptr< Image > buttonSettingsDeselect
- std::unique_ptr< Image > buttonDifficultySelect
- std::unique_ptr< Image > buttonDifficultyDeselect
- std::unique_ptr< Image > buttonLeaderboardSelect
- std::unique_ptr< Image > buttonLeaderboardDeselect
- std::unique_ptr< Image > buttonExitSelect
- std::unique_ptr< Image > buttonExitDeselect

Outros membros herdados

5.10.1 Descrição detalhada

Tela inicial do jogo, centraliza navegação para demais estados.

5.10.2 Construtores e Destrutores

5.10.2.1 MainMenu()

```
MainMenu::MainMenu ( )
```

Construtor da classe MainMenu.

Bibliotecas necessárias

Carrega todos os recursos visuais necessários para a tela do menu, incluindo o logo e as imagens para cada botão nos estados selecionado e não selecionado.

5.10.3 Funções membros

5.10.3.1 draw()

```
void MainMenu::draw (
            Motion & motion )  [override], [virtual]
```

Desenha todos os elementos visuais do menu principal na tela.

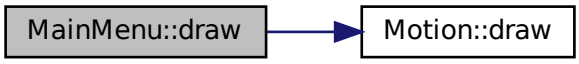
Parâmetros

motion	Referência para desenhar o fundo animado.

Renderiza o logo, o fundo e cada um dos botões com base na interação do usuário.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.10.3.2 enter()

```
void MainMenu::enter ( )  [override], [virtual]
```

Prepara o menu para exibição.

Esta função é chamada toda vez que o MainMenu se torna o estado ativo. Ela garante que o menu seja resetado para um estado inicial consistente, com o primeiro botão selecionado.

Implementa State.

5.10.3.3 handle_input()

```
State * MainMenu::handle_input (
            const ALLEGRO_EVENT & ev )  [override], [virtual]
```

Gerencia a entrada do teclado para navegar e selecionar opções no menu.

Parâmetros

ev	O evento da Allegro a ser processado.

Retorna

Retorna um ponteiro para o próximo estado do jogo, ou nullptr para sair.

Implementa State.

5.10.3.4 update()

```
State * MainMenu::update (
            Motion & motion )  [override], [virtual]
```

Atualiza a lógica do estado do movimento/animação de fundo a cada quadro.

Parâmetros

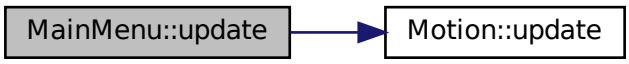
motion	Referência ao objeto de controle de animação do fundo.

Retorna

Retorna 'this' para indicar a permanência no estado atual.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.10.4 Atributos

5.10.4.1 buttonDifficultyDeselect

```
std::unique_ptr<Image> MainMenu::buttonDifficultyDeselect  [private]
```

5.10.4.2 buttonDifficultySelect

```
std::unique_ptr<Image> MainMenu::buttonDifficultySelect  [private]
```

5.10.4.3 buttonExitDeselect

```
std::unique_ptr<Image> MainMenu::buttonExitDeselect  [private]
```

5.10.4.4 buttonExitSelect

```
std::unique_ptr<Image> MainMenu::buttonExitSelect  [private]
```

5.10.4.5 buttonLeaderboardDeselect

```
std::unique_ptr<Image> MainMenu::buttonLeaderboardDeselect  [private]
```

5.10.4.6 buttonLeaderboardSelect

```
std::unique_ptr<Image> MainMenu::buttonLeaderboardSelect  [private]
```

5.10.4.7 buttonLoadGameDeselect

```
std::unique_ptr<Image> MainMenu::buttonLoadGameDeselect  [private]
```

5.10.4.8 buttonLoadGameSelect

```
std::unique_ptr<Image> MainMenu::buttonLoadGameSelect  [private]
```

5.10.4.9 buttonNewGameDeselect

```
std::unique_ptr<Image> MainMenu::buttonNewGameDeselect  [private]
```

5.10.4.10 buttonNewGameSelect

```
std::unique_ptr<Image> MainMenu::buttonNewGameSelect  [private]
```

5.10.4.11 buttonPositionSelected

```
int MainMenu::buttonPositionSelected = 0
```

5.10.4.12 buttonSettingsDeselect

```
std::unique_ptr<Image> MainMenu::buttonSettingsDeselect  [private]
```

5.10.4.13 buttonSettingsSelect

```
std::unique_ptr<Image> MainMenu::buttonSettingsSelect  [private]
```

5.10.4.14 logoNormal

```
std::unique_ptr<Image> MainMenu::logoNormal  [private]
```

Ponteiros inteligentes para todos os recursos visuais (imagens) do menu.

5.10.4.15 menuButtons

```
std::vector<Button> MainMenu::menuButtons
```

Valor inicial:

```
= {{"NewGame", 1},     {"LoadGame", 0},
                                    {"Settings", 0},    {"Difficulty", 0},
                                    {"Leaderboard", 0}, {"Exit", 0}}
```

Vetor que armazena os botões do menu e seus estados de seleção.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/states/main_menu.hpp
- src/states/main_menu.cpp

5.11 Referência da Classe Motion

Responsável por atualizar e desenhar efeitos de parallax, clima e piso animado.

```
#include <motion.hpp>
```

Membros Públicos

- void draw ()

  Desenha todos os elementos do background baseado no cenário ativo.
- void update ()

  Atualiza posições dos elementos de background conforme cenário.
- Motion ()
- void setController (int)

  Seleciona o cenário (1 = dia, 2 = neve, 3 = chuva).

Atributos Privados

- int controller = 1
- float controll = 0
- float contF = 0
- float contB = 400
- float speedRain = 6
- float positionR1 = 400
- float positionR2 = -320
- float speedFlakes = 1.0
- float positionF = 500
- float positionF2 = 550
- float positionF3 = -210
- float speedFlakes2 = 0.4
- float positionL_x = 0
- float speed_light = 0.5
- float positionC_x = 0
- float positionCC_x = -1280
- float speed_cloud = 0.6
- float positionB_x = 0
- float speed_little = 1
- float cont = 16
- float value = 0
- float positionF_x = 1280
- float positionF2_x = 0
- float speed_floor = 2.5
- std::unique_ptr< Image > background_rain
- std::unique_ptr< Image > drips1
- std::unique_ptr< Image > drips2
- std::unique_ptr< Image > thunder
- std::unique_ptr< Image > thunder1
- std::unique_ptr< Image > background_snow
- std::unique_ptr< Image > flakesLittle
- std::unique_ptr< Image > flakesLittle2
- std::unique_ptr< Image > flakesBig
- std::unique_ptr< Image > background
- std::unique_ptr< Image > lights
- std::unique_ptr< Image > clouds
- std::unique_ptr< Image > clouds2
- std::unique_ptr< Image > little
- std::unique_ptr< Image > little2
- std::unique_ptr< Image > little3
- std::unique_ptr< Image > bird1
- std::unique_ptr< Image > bird2
- std::unique_ptr< Image > bird3
- std::unique_ptr< Image > ground
- std::unique_ptr< Image > ground2

5.11.1 Descrição detalhada

Responsável por atualizar e desenhar efeitos de parallax, clima e piso animado.

5.11.2 Construtores e Destrutores

5.11.2.1 Motion()

```
Motion::Motion ( )
```

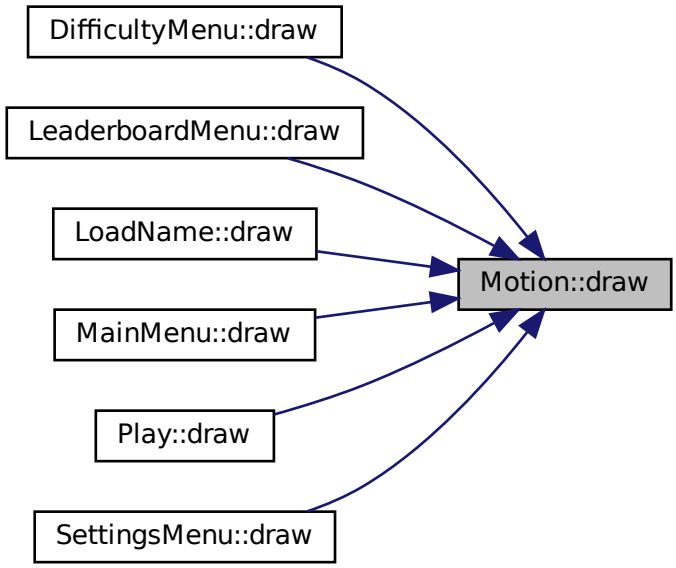
5.11.3 Funções membros

5.11.3.1 draw()

```
void Motion::draw ( )
```

Desenha todos os elementos do background baseado no cenário ativo.

Esse é o diagrama das funções que utilizam essa função:



5.11.3.2 setController()

```
void Motion::setController (
            int x )
```

Seleciona o cenário (1 = dia, 2 = neve, 3 = chuva).

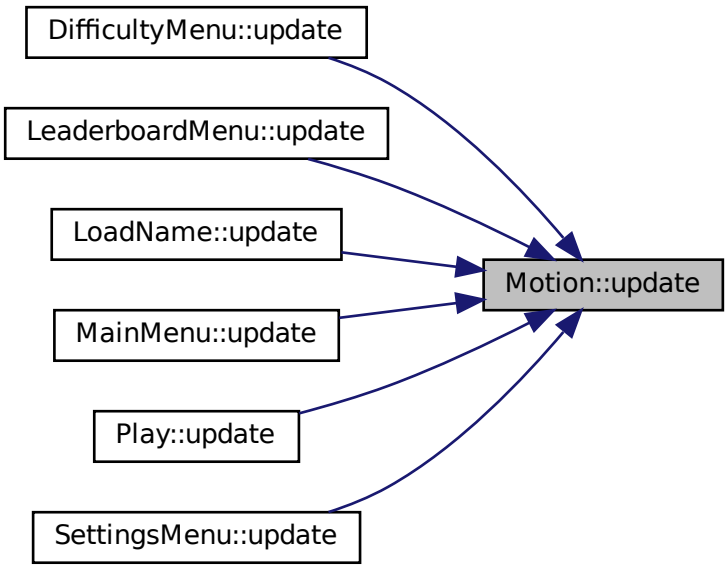
Esse é o diagrama das funções que utilizam essa função:

5.11.3.3 update()

```
void Motion::update ( )
```

Atualiza posições dos elementos de background conforme cenário.

Esse é o diagrama das funções que utilizam essa função:



5.11.4 Atributos

5.11.4.1 background

```
std::unique_ptr<Image> Motion::background  [private]
```

5.11.4.2 background_rain

```
std::unique_ptr<Image> Motion::background_rain  [private]
```

5.11.4.3 background_snow

```
std::unique_ptr<Image> Motion::background_snow  [private]
```

5.11.4.4 bird1

```
std::unique_ptr<Image> Motion::bird1  [private]
```

5.11.4.5 bird2

```
std::unique_ptr<Image> Motion::bird2  [private]
```

5.11.4.6 bird3

```
std::unique_ptr<Image> Motion::bird3  [private]
```

5.11.4.7 clouds

```
std::unique_ptr<Image> Motion::clouds  [private]
```

5.11.4.8 clouds2

```
std::unique_ptr<Image> Motion::clouds2  [private]
```

5.11.4.9 cont

```
float Motion::cont = 16  [private]
```

5.11.4.10 contB

```
float Motion::contB = 400  [private]
```

5.11.4.11 contF

```
float Motion::contF = 0  [private]
```

5.11.4.12 controll

```
float Motion::controll = 0  [private]
```

5.11.4.13 controller

```
int Motion::controller = 1  [private]
```

5.11.4.14 drips1

```
std::unique_ptr<Image> Motion::drips1  [private]
```

5.11.4.15 drips2

```
std::unique_ptr<Image> Motion::drips2  [private]
```

5.11.4.16 flakesBig

```
std::unique_ptr<Image> Motion::flakesBig  [private]
```

5.11.4.17 flakesLittle

```
std::unique_ptr<Image> Motion::flakesLittle  [private]
```

5.11.4.18 flakesLittle2

```
std::unique_ptr<Image> Motion::flakesLittle2  [private]
```

5.11.4.19 ground

```
std::unique_ptr<Image> Motion::ground  [private]
```

5.11.4.20 ground2

```
std::unique_ptr<Image> Motion::ground2  [private]
```

5.11.4.21 lights

```
std::unique_ptr<Image> Motion::lights  [private]
```

5.11.4.22 little

```
std::unique_ptr<Image> Motion::little  [private]
```

5.11.4.23 little2

```
std::unique_ptr<Image> Motion::little2  [private]
```

5.11.4.24 little3

```
std::unique_ptr<Image> Motion::little3  [private]
```

5.11.4.25 positionB_x

```
float Motion::positionB_x = 0  [private]
```

5.11.4.26 positionC_x

```
float Motion::positionC_x = 0  [private]
```

5.11.4.27 positionCC_x

```
float Motion::positionCC_x = -1280  [private]
```

5.11.4.28 positionF

```
float Motion::positionF = 500  [private]
```

5.11.4.29 positionF2

```
float Motion::positionF2 = 550  [private]
```

5.11.4.30 positionF2_x

```
float Motion::positionF2_x = 0  [private]
```

5.11.4.31 positionF3

```
float Motion::positionF3 = -210  [private]
```

5.11.4.32 positionF_x

```
float Motion::positionF_x = 1280  [private]
```

5.11.4.33 positionL_x

```
float Motion::positionL_x = 0  [private]
```

5.11.4.34 positionR1

```
float Motion::positionR1 = 400  [private]
```

5.11.4.35 positionR2

```
float Motion::positionR2 = -320  [private]
```

5.11.4.36 speed_cloud

```
float Motion::speed_cloud = 0.6  [private]
```

5.11.4.37 speed_floor

```
float Motion::speed_floor = 2.5  [private]
```

5.11.4.38 speed_light

```
float Motion::speed_light = 0.5  [private]
```

5.11.4.39 speed_little

```
float Motion::speed_little = 1  [private]
```

5.11.4.40 speedFlakes

```
float Motion::speedFlakes = 1.0  [private]
```

5.11.4.41 speedFlakes2

```
float Motion::speedFlakes2 = 0.4  [private]
```

5.11.4.42 speedRain

```
float Motion::speedRain = 6  [private]
```

5.11.4.43 thunder

```
std::unique_ptr<Image> Motion::thunder  [private]
```

5.11.4.44 thunder1

```
std::unique_ptr<Image> Motion::thunder1  [private]
```

5.11.4.45 value

```
float Motion::value = 0  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/motion.hpp
- src/motion.cpp

5.12 Referência da Classe Music

Gerencia reprodução em loop de faixas de áudio.

```
#include <assets.hpp>
```

Membros Públicos

- Music (const char ∗diretorio)

  Constrói um objeto de música em loop.
- void playMusic ()

  Inicia a reprodução da música.
- ∼Music ()

  Libera o stream de áudio.

Atributos Privados

- ALLEGRO_AUDIO_STREAM ∗ music = NULL

5.12.1 Descrição detalhada

Gerencia reprodução em loop de faixas de áudio.

5.12.2 Construtores e Destrutores

5.12.2.1 Music()

```
Music::Music (
            const char * diretorio )
```

Constrói um objeto de música em loop.

Parâmetros

diretorio	Caminho do arquivo de áudio.

Exceções

std::bad_alloc	Se falhar ao carregar o stream.

5.12.2.2 ∼Music()

```
Music::~Music ( )
```

Libera o stream de áudio.

5.12.3 Funções membros

5.12.3.1 playMusic()

```
void Music::playMusic ( )
```

Inicia a reprodução da música.

5.12.4 Atributos

5.12.4.1 music

```
ALLEGRO_AUDIO_STREAM* Music::music = NULL  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/assets.hpp
- src/assets.cpp

5.13 Referência da Classe Pipe

Obstáculo individual que se move horizontalmente e gera pontuação.

```
#include <pipe.hpp>
```

Diagrama de hierarquia para Pipe:

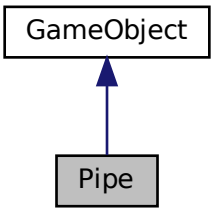
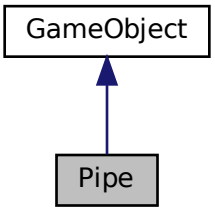


Diagrama de colaboração para Pipe:



Membros Públicos

- Pipe (ALLEGRO_BITMAP ∗img, float x, float y)

  Construct a pipe object.
- void update () override

  Move the pipe horizontally and update bounding box.
- void draw () override

  Draw the pipe sprite at its current position.
- bool is_off_screen () const

  Check whether the pipe has left the visible screen.
- bool check_score (float bird_x)

  Evaluate score condition for this pipe.
- void set_vx (float vel)

  Set horizontal velocity.
- void set_y (float new_y)

  Force a new y position (used for vertical movement).
- float get_y ()

  Get current y position.

Atributos Privados

- float vx = 0

  horizontal velocity (pixels per frame)
- bool scored

  true after the bird has already scored on this pipe

Outros membros herdados

5.13.1 Descrição detalhada

Obstáculo individual que se move horizontalmente e gera pontuação.

5.13.2 Construtores e Destrutores

5.13.2.1 Pipe()

```
Pipe::Pipe (
            ALLEGRO_BITMAP * img,
            float x,
            float y )  [inline]
```

Construct a pipe object.

Parâmetros

img	Pointer to loaded bitmap.
x	Initial x position.
y	Initial y position.

Initializes GameObject base, sets velocity and score flag. Este é o diagrama das funções utilizadas por essa função:

5.13.3 Funções membros

5.13.3.1 check_score()

```
bool Pipe::check_score (
            float bird_x )
```

Evaluate score condition for this pipe.

Gera pontuação quando o pássaro passa o cano.

Parâmetros

bird↩ _x	x position of the bird.

Retorna

true if the bird just passed the pipe for the first time.

Parâmetros

bird_x	Coordenada X do pássaro.

5.13.3.2 draw()

```
void Pipe::draw ( )  [override], [virtual]
```

Draw the pipe sprite at its current position.

Desenha o cano.

Implementa GameObject.

5.13.3.3 get_y()

```
float Pipe::get_y ( )
```

Get current y position.

Retorna posição Y atual.

5.13.3.4 is_off_screen()

```
bool Pipe::is_off_screen ( ) const
```

Check whether the pipe has left the visible screen.

Verifica se o cano saiu da tela pela esquerda.

5.13.3.5 set_vx()

```
void Pipe::set_vx (
            float vel )
```

Set horizontal velocity.

Define velocidade horizontal dos canos.

5.13.3.6 set_y()

```
void Pipe::set_y (
            float new_y )
```

Force a new y position (used for vertical movement).

Força nova posição Y.

5.13.3.7 update()

```
void Pipe::update ( )  [override], [virtual]
```

Move the pipe horizontally and update bounding box.

Atualiza posição horizontal do cano.

Implementa GameObject.

Este é o diagrama das funções utilizadas por essa função:

5.13.4 Atributos

5.13.4.1 scored

```
bool Pipe::scored  [private]
```

true after the bird has already scored on this pipe

5.13.4.2 vx

```
float Pipe::vx = 0  [private]
```

horizontal velocity (pixels per frame)

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/pipe.hpp
- src/pipe.cpp

5.14 Referência da Classe PipeList

Gerencia vetor de PipePair: spawn, atualização, colisão e score.

`#include <pipe.hpp>`

Membros Públicos

- PipeList (ALLEGRO_BITMAP *img)
  Construct a list manager.
- void draw ()
  Draw every pipe pair.
- void set_vx (float vel)
  Set uniform horizontal velocity for all pipes.
- void update ()
  Update positions of all pipes and handle vertical motion.
- void add_pipe_pair ()
  Spawn a new pair if gap condition is satisfied.
- void delete_pipe_pair ()
  Remove pipes that left the screen.
- bool check_collision (GameObject &bird)
  Check bird collision against any pipe.
- bool check_score (float bird_x)
  Check if any pipe pair increments the score.
- bool set_start ()
  Begin spawning logic (first call after game starts).
- int get_points ()
  Get current score.
- void set_difficulty (int diff)
  Configure difficulty (1 = static, 2 = vertical movers).
- const std::vector< PipePair > & get_pipe_pairs () const
  Expose internal pipe vector (read-only).
- void reset ()
  Clear list and reset all counters.

Atributos Privados

- ALLEGRO_BITMAP * pipe1 = nullptr
  bitmap reference for new pipes
- std::vector< PipePair > Pipes
  active pipe pairs
- bool start = false
  true after first pipe is spawned
- int points = 0
  current score
- int difficulty_pipe = 1
  1 = static, 2 = with movers

Atributos Privados Estáticos

- static std::mt19937 gen {std::random_device{}()}

5.14.1 Descrição detalhada

Gerencia vetor de PipePair: spawn, atualização, colisão e score.

5.14.2 Construtores e Destrutores

5.14.2.1 PipeList()

```
PipeList::PipeList (
            ALLEGRO_BITMAP * img )
```

Construct a list manager.

Parâmetros

img	Bitmap to use when spawning new pipe pairs.

5.14.3 Funções membros

5.14.3.1 add_pipe_pair()

```
void PipeList::add_pipe_pair ( )
```

Spawn a new pair if gap condition is satisfied.

Spawna novo par de canos conforme distância do último. Esse é o diagrama das funções que utilizam essa função:

5.14.3.2 check_collision()

```
bool PipeList::check_collision (
            GameObject & bird )
```

Check bird collision against any pipe.

Verifica colisão do pássaro com qualquer cano ativo.

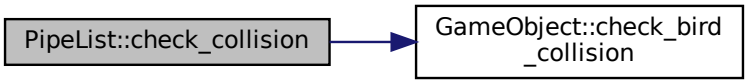
Parâmetros

bird	Player object (as GameObject).

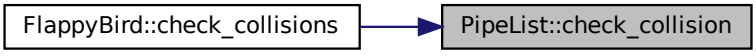
Retorna

true if a collision is detected.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



5.14.3.3 check_score()

```
bool PipeList::check_score (
            float bird_x )
```

Check if any pipe pair increments the score.

Atualiza pontuação se o pássaro ultrapassar algum cano.

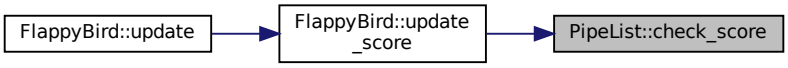
Parâmetros

bird↩ _x	Bird x coordinate.

Retorna

true if score increases.

Esse é o diagrama das funções que utilizam essa função:



5.14.3.4 delete_pipe_pair()

```
void PipeList::delete_pipe_pair ( )
```

Remove pipes that left the screen.

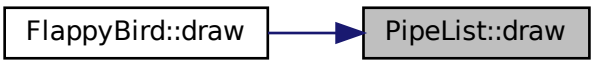
Remove canos que saíram da tela. Esse é o diagrama das funções que utilizam essa função:

5.14.3.5 draw()

```
void PipeList::draw ( )
```

Draw every pipe pair.

Esse é o diagrama das funções que utilizam essa função:



5.14.3.6 get_pipe_pairs()

```
const std::vector<PipePair>& PipeList::get_pipe_pairs ( ) const  [inline]
```

Expose internal pipe vector (read-only).

5.14.3.7 get_points()

```
int PipeList::get_points ( )
```

Get current score.

Esse é o diagrama das funções que utilizam essa função:

5.14.3.8 reset()

```
void PipeList::reset ( )
```

Clear list and reset all counters.

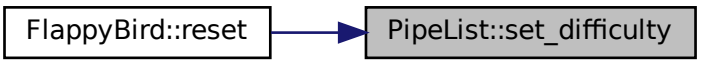
Limpa lista de canos e zera pontos. Esse é o diagrama das funções que utilizam essa função:

5.14.3.9 set_difficulty()

```
void PipeList::set_difficulty (
            int diff )
```

Configure difficulty (1 = static, 2 = vertical movers).

Define dificuldade (1 estático, 2 com movimento vertical). Esse é o diagrama das funções que utilizam essa função:



5.14.3.10 set_start()

```
bool PipeList::set_start ( )
```

Begin spawning logic (first call after game starts).

Ativa geração de canos na primeira chamada.

Retorna

true on first activation, false otherwise.

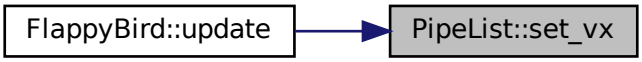
Esse é o diagrama das funções que utilizam essa função:

5.14.3.11 set_vx()

```
void PipeList::set_vx (
            float vel )
```

Set uniform horizontal velocity for all pipes.

Esse é o diagrama das funções que utilizam essa função:



5.14.3.12 update()

```
void PipeList::update ( )
```

Update positions of all pipes and handle vertical motion.

Esse é o diagrama das funções que utilizam essa função:

5.14.4 Atributos

5.14.4.1 difficulty_pipe

```
int PipeList::difficulty_pipe = 1  [private]
```

1 = static, 2 = with movers

5.14.4.2 gen

```
std::mt19937 PipeList::gen {std::random_device{}()}  [inline], [static], [private]
```

5.14.4.3 pipe1

```
ALLEGRO_BITMAP* PipeList::pipe1 = nullptr  [private]
```

bitmap reference for new pipes

5.14.4.4 Pipes

```
std::vector<PipePair> PipeList::Pipes  [private]
```

active pipe pairs

5.14.4.5 points

```
int PipeList::points = 0  [private]
```

current score

5.14.4.6 start

```
bool PipeList::start = false  [private]
```

true after first pipe is spawned

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

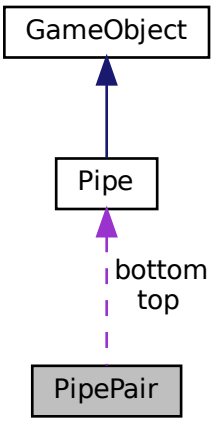
- include/pipe.hpp
- src/pipe.cpp

5.15 Referência da Classe PipePair

Agrupa dois canos (superior e inferior) podendo ter movimento vertical.

`#include <pipe.hpp>`

Diagrama de colaboração para PipePair:



Membros Públicos

- PipePair (ALLEGRO_BITMAP ∗img, float x, float y)
  Create a static pair of pipes.
- PipePair (ALLEGRO_BITMAP ∗img, float x, float y, bool type, bool direction)
  Create a pair that can optionally move up/down.

Atributos Públicos

- Pipe bottom
  lower pipe
- Pipe top
  upper pipe
- bool movement = false
  true if pair moves vertically
- float signal = 1.0
  direction multiplier for vertical motion

5.15.1 Descrição detalhada

Agrupa dois canos (superior e inferior) podendo ter movimento vertical.

5.15.2 Construtores e Destrutores

5.15.2.1 PipePair() [1/2]

```
PipePair::PipePair (
            ALLEGRO_BITMAP * img,
            float x,
            float y )  [inline]
```

Create a static pair of pipes.

Parâmetros

img	Bitmap shared by both pipes.
x	Initial x coordinate.
y	Bottom pipe y coordinate.

5.15.2.2 PipePair() [2/2]

```
PipePair::PipePair (
            ALLEGRO_BITMAP * img,
            float x,
            float y,
            bool type,
            bool direction )  [inline]
```

Create a pair that can optionally move up/down.

Parâmetros

img	Bitmap shared by both pipes.
x	Initial x coordinate.
y	Bottom pipe y coordinate.
type	Enable vertical movement.
direction	Initial movement direction (true = upward).

5.15.3 Atributos

5.15.3.1 bottom

```
Pipe PipePair::bottom
```

lower pipe

5.15.3.2 movement

```
bool PipePair::movement = false
```

true if pair moves vertically

5.15.3.3 signal

```
float PipePair::signal = 1.0
```

direction multiplier for vertical motion

5.15.3.4 top

```
Pipe PipePair::top
```

upper pipe

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- include/pipe.hpp

5.16 Referência da Classe Play

Estado principal de jogo: controla loop ativo, pontuação e game over.

```
#include <play.hpp>
```

Diagrama de hierarquia para Play:

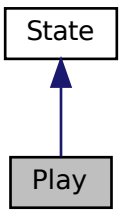
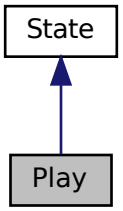


Diagrama de colaboração para Play:



Membros Públicos

- void enter () override

  Inicia ou reinicia a partida.
- State ∗ handle_input (const ALLEGRO_EVENT &ev) override

  Processa entrada do usuario.
- State ∗ update (Motion &motion) override

  Atualiza simulacao do jogo.
- void draw (Motion &motion) override

  Renderiza cena completa.
- Play ()

  Construtor padrao.

Atributos Privados

- std::unique_ptr< FlappyBird > flappy
- ScreenState status
- std::unique_ptr< Image > logoGameOver
- std::unique_ptr< Image > buttonTryagainSelect
- std::unique_ptr< Image > buttonTryagainDeselect
- std::unique_ptr< Image > buttonExitSelect
- std::unique_ptr< Image > buttonExitDeselect
- std::unique_ptr< Image > buttonPause
- std::unique_ptr< TextFont > font
- int buttonPositionSelected = 0
- std::vector< Button > menuButtons

Outros membros herdados

5.16.1 Descrição detalhada

Estado principal de jogo: controla loop ativo, pontuação e game over.

5.16.2 Construtores e Destrutores

5.16.2.1 Play()

```
Play::Play ( )
```

Construtor padrao.

Carrega todos os bitmaps, botoes e fontes exigidos durante o estado de jogo.

5.16.3 Funções membros

5.16.3.1 draw()

```
void Play::draw (
            Motion & motion )  [override], [virtual]
```

Renderiza cena completa.

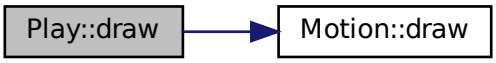
Desenha fundo, personagem, menus de pausa ou game over conforme o estado em curso.

Parâmetros

motion	Sistema de camera/transformacoes usado na renderizacao.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.16.3.2 enter()

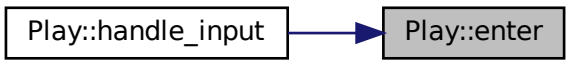
```
void Play::enter ( )  [override], [virtual]
```

Inicia ou reinicia a partida.

Cria um novo FlappyBird, limpa pontuacao e define status PLAY.

Implementa State.

Esse é o diagrama das funções que utilizam essa função:



5.16.3.3 handle_input()

```
State * Play::handle_input (
            const ALLEGRO_EVENT & ev )  [override], [virtual]
```

Processa entrada do usuario.

Mapeia teclas para pulo, pausa, retomada e menu de game over.

Parâmetros

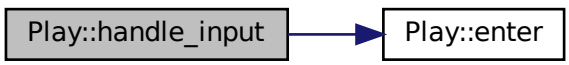
ev	Evento Allegro recebido da fila.

Retorna

Ponteiro para o estado resultante apos o processamento.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.16.3.4 update()

```
State * Play::update (
            Motion & motion )  [override], [virtual]
```

Atualiza simulacao do jogo.

Avanca objetos quando status eh PLAY e detecta colisoes para transicionar a GAME_OVER.

Parâmetros

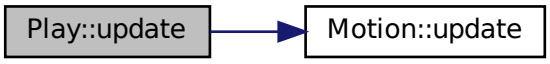
motion	Sistema de camera/transformacoes para parallax ou shake.

Retorna

Ponteiro para o estado atual para continuidade.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.16.4 Atributos

5.16.4.1 buttonExitDeselect

```
std::unique_ptr<Image> Play::buttonExitDeselect  [private]
```

5.16.4.2 buttonExitSelect

```
std::unique_ptr<Image> Play::buttonExitSelect  [private]
```

5.16.4.3 buttonPause

```
std::unique_ptr<Image> Play::buttonPause  [private]
```

5.16.4.4 buttonPositionSelected

```
int Play::buttonPositionSelected = 0  [private]
```

5.16.4.5 buttonTryagainDeselect

```
std::unique_ptr<Image> Play::buttonTryagainDeselect  [private]
```

5.16.4.6 buttonTryagainSelect

```
std::unique_ptr<Image> Play::buttonTryagainSelect  [private]
```

5.16.4.7 flappy

```
std::unique_ptr<FlappyBird> Play::flappy  [private]
```

5.16.4.8 font

```
std::unique_ptr<TextFont> Play::font  [private]
```

5.16.4.9 logoGameOver

```
std::unique_ptr<Image> Play::logoGameOver  [private]
```

5.16.4.10 menuButtons

```
std::vector<Button> Play::menuButtons  [private]
```

Valor inicial:

```
= {
      {"TryAgain", 1}, {"Exit", 0}}
```

5.16.4.11 status

```
ScreenState Play::status  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/states/play.hpp
- src/states/play.cpp

5.17 Referência da Classe Player

Armazena nome e pontuação e provê utilidades para ranking.

```
#include <player.hpp>
```

Membros Públicos

- Player (std::string name, int score=0)

  Constrói um jogador com nome e pontuação inicial.
- std::string GetName () const

  Retorna o nome do jogador.
- int GetScore () const

  Retorna a pontuação atual.
- void SetScore (int points)

  Atualiza (sobrescreve) a pontuação do jogador.
- bool operator< (const Player &other_p) const

  Define critério de ordenação decrescente por pontuação.

Membros Públicos Estáticos

- static bool CheckingName (std::vector< Player > &ranking, std::string &Name)

  Verifica se um nome já existe no ranking.
- static std::vector< Player > ReadLeaderboard (std::string file_name)

  Lê arquivo de ranking e retorna vetor de jogadores.
- static void SortLeaderboard (std::vector< Player > &ranking)

  Ordena vetor de jogadores por pontuação decrescente.
- static void SaveLeaderboard (std::string fileName, std::vector< Player > &ranking)

  Salva lista de jogadores no arquivo, sobrescrevendo conteúdo.
- static void SaveLeaderboard (std::string fileName, std::vector< Player > &ranking, const Player &new_↩ player)

  Salva ranking adicionando ou atualizando um jogador.
- static void ShowLeaderboard (std::vector< Player > &ranking, ALLEGRO_FONT *font)

Atributos Privados

- std::string name
- int score

5.17.1 Descrição detalhada

Armazena nome e pontuação e provê utilidades para ranking.

5.17.2 Construtores e Destrutores

5.17.2.1 Player()

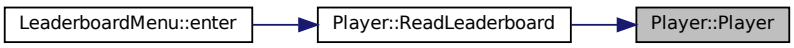
```
Player::Player (
            std::string name,
            int score = 0 )
```

Constrói um jogador com nome e pontuação inicial.

Parâmetros

name	Nome do jogador.
score	Pontuação inicial (0 por padrão).

Esse é o diagrama das funções que utilizam essa função:



5.17.3 Funções membros

5.17.3.1 CheckingName()

```
bool Player::CheckingName (
            std::vector< Player > & ranking,
            std::string & Name )  [static]
```

Verifica se um nome já existe no ranking.

Parâmetros

ranking	Vetor de jogadores.
Name	Nome a procurar.

Retorna

true se encontrado.

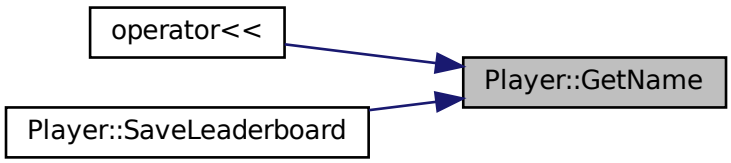
Esse é o diagrama das funções que utilizam essa função:

5.17.3.2 GetName()

```
std::string Player::GetName ( ) const
```

Retorna o nome do jogador.

Esse é o diagrama das funções que utilizam essa função:

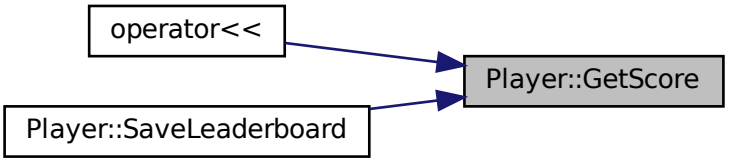


5.17.3.3 GetScore()

```
int Player::GetScore ( ) const
```

Retorna a pontuação atual.

Esse é o diagrama das funções que utilizam essa função:



5.17.3.4 operator<()

```
bool Player::operator< (
            const Player & other_p ) const
```

Define critério de ordenação decrescente por pontuação.

5.17.3.5 ReadLeaderboard()

```
std::vector< Player > Player::ReadLeaderboard (
            std::string file_name )  [static]
```

Lê arquivo de ranking e retorna vetor de jogadores.

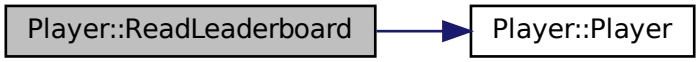
Parâmetros

file_name	Caminho do arquivo.

Retorna

Vetor de jogadores lido.

Este é o diagrama das funções utilizadas por essa função:



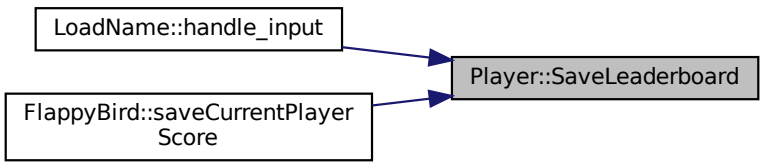
Esse é o diagrama das funções que utilizam essa função:

5.17.3.6 SaveLeaderboard() [1/2]

```
void Player::SaveLeaderboard (
            std::string fileName,
            std::vector< Player > & ranking )  [static]
```

Salva lista de jogadores no arquivo, sobrescrevendo conteúdo.

Esse é o diagrama das funções que utilizam essa função:



5.17.3.7 SaveLeaderboard() [2/2]

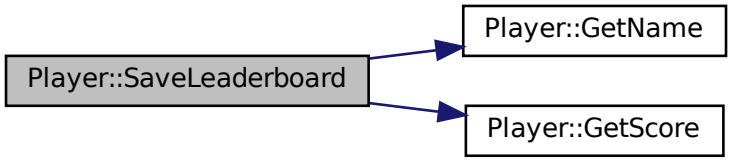
```
void Player::SaveLeaderboard (
            std::string fileName,
            std::vector< Player > & ranking,
            const Player & new_player )  [static]
```

Salva ranking adicionando ou atualizando um jogador.

Parâmetros

new_player	Jogador a inserir/atualizar.

Este é o diagrama das funções utilizadas por essa função:



5.17.3.8 SetScore()

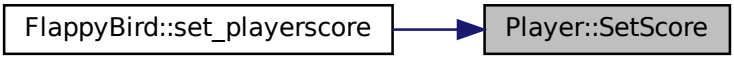
```
void Player::SetScore (
            int points )
```

Atualiza (sobrescreve) a pontuação do jogador.

Parâmetros

points	Novo valor de pontuação.

Esse é o diagrama das funções que utilizam essa função:



5.17.3.9 ShowLeaderboard()

```
static void Player::ShowLeaderboard (
            std::vector< Player > & ranking,
            ALLEGRO_FONT * font )  [static]
```

5.17.3.10 SortLeaderboard()

```
void Player::SortLeaderboard (
            std::vector< Player > & ranking )  [static]
```

Ordena vetor de jogadores por pontuação decrescente.

Esse é o diagrama das funções que utilizam essa função:

5.17.4 Atributos

5.17.4.1 name

```
std::string Player::name  [private]
```

5.17.4.2 score

```
int Player::score  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/player.hpp
- src/player.cpp

5.18 Referência da Classe SettingsMenu

Tela de configurações: permite alterar clima e habilitar/desabilitar música.

`#include <settings_menu.hpp>`

Diagrama de hierarquia para SettingsMenu:

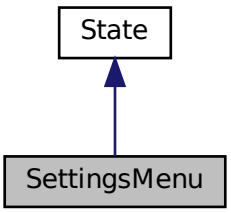
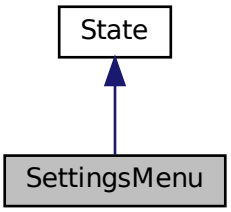


Diagrama de colaboração para SettingsMenu:



Membros Públicos

- State * handle_input (const ALLEGRO_EVENT &ev) override

  Processa a entrada do usuário (teclado e mouse).
- State * update (Motion &motion) override

  Atualiza a lógica do estado de movimento/animação de fundo.
- void draw (Motion &motion) override

  Desenha todos os elementos visuais do estado na tela.
- void enter () override

  Executa ações ao entrar neste estado.
- SettingsMenu ()

  Construtor da classe SettingsMenu.

Atributos Públicos

- int buttonPositionSelected = 0
- std::string weatherSelected = "noOne"
- bool musicState = true
- std::vector< Button > menuButtons

  Vetor que armazena os botões do menu e seus estados de seleção.

Atributos Privados

- std::unique_ptr< Image > campSettingsMusic

  Ponteiros inteligentes para todos os recursos visuais (imagens e fontes) do menu.
- std::unique_ptr< Image > campSettingsNoMusic
- std::unique_ptr< Image > buttonMusicSelect
- std::unique_ptr< Image > buttonMusicDaySelect
- std::unique_ptr< Image > buttonMusicRainSelect
- std::unique_ptr< Image > buttonMusicSnowSelect
- std::unique_ptr< Image > buttonNoMusicDaySelect
- std::unique_ptr< Image > buttonNoMusicRainSelect
- std::unique_ptr< Image > buttonNoMusicSnowSelect
- std::unique_ptr< Image > buttonBackSelect
- std::unique_ptr< Image > buttonBackDeselect
- std::unique_ptr< TextFont > font

Outros membros herdados

5.18.1 Descrição detalhada

Tela de configurações: permite alterar clima e habilitar/desabilitar música.

Bibliotecas necessárias

5.18.2 Construtores e Destrutores

5.18.2.1 SettingsMenu()

```
SettingsMenu::SettingsMenu ( )
```

Construtor da classe SettingsMenu.

Bibliotecas necessárias

Inicializa e carrega todos os recursos gráficos (imagens e fontes) necessários para a tela de configurações.

5.18.3 Funções membros

5.18.3.1 draw()

```
void SettingsMenu::draw (
            Motion & motion )  [override], [virtual]
```

Desenha todos os elementos visuais do estado na tela.

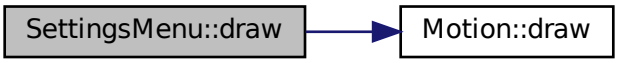
Parâmetros

motion	Referência ao objeto de controle de movimento para desenhar o fundo.

A função renderiza o fundo, os botões e os textos informativos com base no estado atual das configurações (música ligada/desligada, botão selecionado).

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.18.3.2 enter()

```
void SettingsMenu::enter ( )  [override], [virtual]
```

Executa ações ao entrar neste estado.

Garante que o menu de configurações seja inicializado em um estado visual previsível, com o primeiro botão ("Música") selecionado por padrão.

Implementa State.

5.18.3.3 handle_input()

```
State * SettingsMenu::handle_input (
            const ALLEGRO_EVENT & ev )  [override], [virtual]
```

Processa a entrada do usuário (teclado e mouse).

Parâmetros

ev	O evento da Allegro a ser processado.

Retorna

Retorna um ponteiro para o próximo estado do jogo. Retorna `this` para continuar neste estado, `nullptr` para fechar a aplicação, ou um novo estado para transição.

Implementa State.

5.18.3.4 update()

```
State * SettingsMenu::update (
            Motion & motion )  [override], [virtual]
```

Atualiza a lógica do estado de movimento/animação de fundo.

Parâmetros

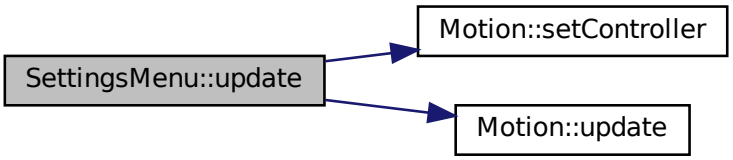
motion	Referência ao objeto de controle de movimento/animação do fundo.

Retorna

Retorna `this` para indicar a permanência no estado atual.

Implementa State.

Este é o diagrama das funções utilizadas por essa função:



5.18.4 Atributos

5.18.4.1 buttonBackDeselect

```
std::unique_ptr<Image> SettingsMenu::buttonBackDeselect  [private]
```

5.18.4.2 buttonBackSelect

```
std::unique_ptr<Image> SettingsMenu::buttonBackSelect  [private]
```

5.18.4.3 buttonMusicDaySelect

```
std::unique_ptr<Image> SettingsMenu::buttonMusicDaySelect  [private]
```

5.18.4.4 buttonMusicRainSelect

```
std::unique_ptr<Image> SettingsMenu::buttonMusicRainSelect  [private]
```

5.18.4.5 buttonMusicSelect

```
std::unique_ptr<Image> SettingsMenu::buttonMusicSelect  [private]
```

5.18.4.6 buttonMusicSnowSelect

```
std::unique_ptr<Image> SettingsMenu::buttonMusicSnowSelect  [private]
```

5.18.4.7 buttonNoMusicDaySelect

```
std::unique_ptr<Image> SettingsMenu::buttonNoMusicDaySelect  [private]
```

5.18.4.8 buttonNoMusicRainSelect

```
std::unique_ptr<Image> SettingsMenu::buttonNoMusicRainSelect  [private]
```

5.18.4.9 buttonNoMusicSnowSelect

```
std::unique_ptr<Image> SettingsMenu::buttonNoMusicSnowSelect  [private]
```

5.18.4.10 buttonPositionSelected

```
int SettingsMenu::buttonPositionSelected = 0
```

5.18.4.11 campSettingsMusic

```
std::unique_ptr<Image> SettingsMenu::campSettingsMusic  [private]
```

Ponteiros inteligentes para todos os recursos visuais (imagens e fontes) do menu.

5.18.4.12 campSettingsNoMusic

```
std::unique_ptr<Image> SettingsMenu::campSettingsNoMusic  [private]
```

5.18.4.13 font

```
std::unique_ptr<TextFont> SettingsMenu::font  [private]
```

5.18.4.14 menuButtons

```
std::vector<Button> SettingsMenu::menuButtons
```

Valor inicial:

```
= {
      {"Music", 1}, {"Day", 0}, {"Snow", 0}, {"Rain", 0},{"Back", 0}}
```

Vetor que armazena os botões do menu e seus estados de seleção.

5.18.4.15 musicState

```
bool SettingsMenu::musicState = true
```

5.18.4.16 weatherSelected

```
std::string SettingsMenu::weatherSelected = "noOne"
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/states/settings_menu.hpp
- src/states/settings_menu.cpp

5.19 Referência da Classe Sound

Carrega e reproduz samples de efeito sonoro.

```
#include <assets.hpp>
```

Membros Públicos

- Sound (const char *diretorio)
  Carrega um efeito sonoro.
- void playSound (float volume)
  Reproduz o efeito sonoro uma única vez.
- ~Sound ()
  Libera o sample de áudio.

Atributos Privados

- ALLEGRO_SAMPLE * sound = NULL

5.19.1 Descrição detalhada

Carrega e reproduz samples de efeito sonoro.

5.19.2 Construtores e Destrutores

5.19.2.1 Sound()

```
Sound::Sound (
            const char * diretorio )
```

Carrega um efeito sonoro.

Parâmetros

diretorio	Caminho do arquivo de som.

Exceções

std::bad_alloc	Caso o sample não seja carregado.

5.19.2.2 ~Sound()

```
Sound::~Sound ( )
```

Libera o sample de áudio.

5.19.3 Funções membros

5.19.3.1 playSound()

```
void Sound::playSound (
            float volume )
```

Reproduz o efeito sonoro uma única vez.

Parâmetros

volume	Volume de 0.0 a 1.0.

5.19.4 Atributos

5.19.4.1 sound

```
ALLEGRO_SAMPLE* Sound::sound = NULL  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

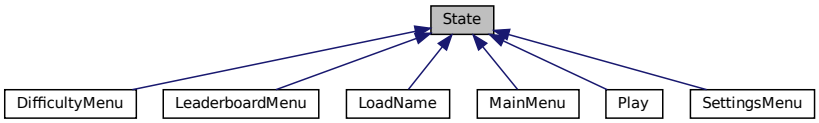
- include/assets.hpp
- src/assets.cpp

5.20 Referência da Classe State

Define a interface que cada tela/estado deve implementar.

`#include <state.hpp>`

Diagrama de hierarquia para State:



Membros Públicos

- virtual ~State ()=default
- virtual void draw (Motion &motion)=0
- virtual State * update (Motion &motion)=0
- virtual State * handle_input (const ALLEGRO_EVENT &ev)=0
- virtual void enter ()=0

Membros Públicos Estáticos

- static void setGlobals (ALLEGRO_DISPLAY *d, ALLEGRO_EVENT_QUEUE *q)

  Configura ponteiros globais de display e fila de eventos.

Atributos Estáticos Protegidos

- static ALLEGRO_DISPLAY * display = nullptr
- static ALLEGRO_EVENT_QUEUE * queue = nullptr
- static ALLEGRO_EVENT ev

5.20.1 Descrição detalhada

Define a interface que cada tela/estado deve implementar.

5.20.2 Construtores e Destrutores

5.20.2.1 ~State()

`virtual State::~State ( )  [virtual], [default]`

5.20.3 Funções membros

5.20.3.1 draw()

```
virtual void State::draw (
            Motion & motion )  [pure virtual]
```

Implementado por SettingsMenu, Play, MainMenu, LoadName, LeaderboardMenu e DifficultyMenu.

5.20.3.2 enter()

```
virtual void State::enter ( )  [pure virtual]
```

Implementado por SettingsMenu, Play, MainMenu, LoadName, LeaderboardMenu e DifficultyMenu.

5.20.3.3 handle_input()

```
virtual State* State::handle_input (
            const ALLEGRO_EVENT & ev )  [pure virtual]
```

Implementado por SettingsMenu, Play, MainMenu, LoadName, LeaderboardMenu e DifficultyMenu.

5.20.3.4 setGlobals()

```
void State::setGlobals (
            ALLEGRO_DISPLAY * d,
            ALLEGRO_EVENT_QUEUE * q )  [static]
```

Configura ponteiros globais de display e fila de eventos.

Esse é o diagrama das funções que utilizam essa função:

5.20.3.5 update()

```
virtual State* State::update (
            Motion & motion )  [pure virtual]
```

Implementado por SettingsMenu, Play, MainMenu, LoadName, LeaderboardMenu e DifficultyMenu.

5.20.4 Atributos

5.20.4.1 display

```
ALLEGRO_DISPLAY * State::display = nullptr  [static], [protected]
```

5.20.4.2 ev

```
ALLEGRO_EVENT State::ev  [static], [protected]
```

5.20.4.3 queue

```
ALLEGRO_EVENT_QUEUE * State::queue = nullptr  [static], [protected]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/state.hpp
- src/state.cpp

5.21 Referência da Classe TextFont

Wrapper para fontes bitmap/TTF e texto colorido.

```
#include <assets.hpp>
```

Membros Públicos

- TextFont (const char *diretorio, int size)

  Carrega uma fonte bitmap.
- void setColor (int r, int g, int b)

  Define a cor utilizada nos próximos desenhos de texto.
- void writeText (const char *texto, int alinhamento, float x, float y)

  Desenha texto na tela.
- ~TextFont ()

  Libera a fonte carregada.

Atributos Privados

- ALLEGRO_FONT * font = NULL
- int r
- int g
- int b = 0

5.21.1 Descrição detalhada

Wrapper para fontes bitmap/TTF e texto colorido.

5.21.2 Construtores e Destrutores

5.21.2.1 TextFont()

```
TextFont::TextFont (
            const char * diretorio,
            int size )
```

Carrega uma fonte bitmap.

Parâmetros

diretorio	Caminho para o arquivo de fonte.
size	Tamanho em pontos.

Exceções

std::bad_alloc	Se a fonte não puder ser carregada.

5.21.2.2 ~TextFont()

```
TextFont::~TextFont ( )
```

Libera a fonte carregada.

5.21.3 Funções membros

5.21.3.1 setColor()

```
void TextFont::setColor (
            int r,
            int g,
            int b )
```

Define a cor utilizada nos próximos desenhos de texto.

5.21.3.2 writeText()

```
void TextFont::writeText (
            const char * texto,
            int alinhamento,
            float x,
            float y )
```

Desenha texto na tela.

Parâmetros

texto	Conteúdo a ser escrito.
alinhamento	ALLEGRO_ALIGN_LEFT/CENTRE/RIGHT.
x	Coordenada X.
y	Coordenada Y.

5.21.4 Atributos

5.21.4.1 b

```
int TextFont::b = 0  [private]
```

5.21.4.2 font

```
ALLEGRO_FONT* TextFont::font = NULL  [private]
```

5.21.4.3 g

```
int TextFont::g  [private]
```

5.21.4.4 r

```
int TextFont::r  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/assets.hpp
- src/assets.cpp

Capítulo 6

Arquivos

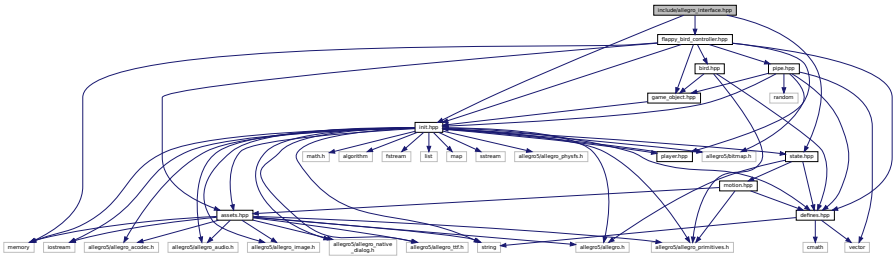
6.1 Referência do Arquivo docs/mainpage.dox

6.2 Referência do Arquivo include/allegro_interface.hpp

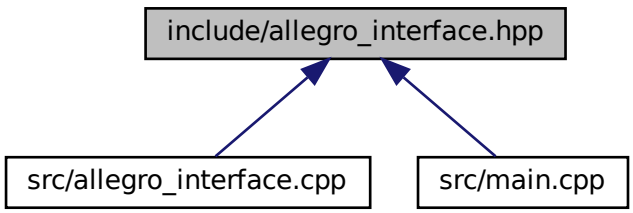
Declara o controlador que integra Allegro e a máquina de estados.

```
#include "flappy_bird_controller.hpp"
#include "init.hpp"
#include "state.hpp"
```

Gráfico de dependência de inclusões para allegro_interface.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class AllegroController

  Gerencia display, eventos e transições de estados do jogo.

6.2.1 Descrição detalhada

Declara o controlador que integra Allegro e a máquina de estados.

Responsável por instanciar `FlappyBird`, gerir fila de eventos Allegro e realizar o loop principal de execução.

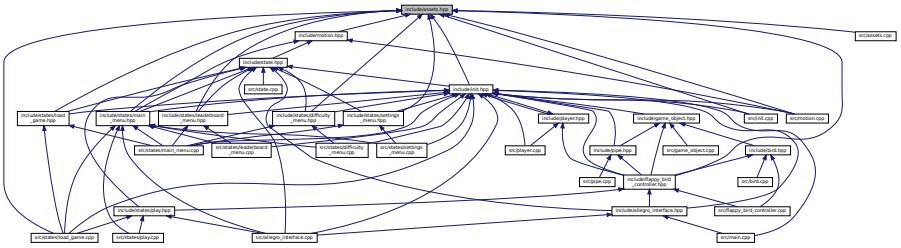
6.3 Referência do Arquivo include/assets.hpp

Declara classes utilitárias para carregar e usar imagens, sons e fontes no Allegro.

```
#include <iostream>
#include <memory>
#include <string>
#include <allegro5/allegro.h>
#include <allegro5/allegro_acodec.h>
#include <allegro5/allegro_audio.h>
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_native_dialog.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_ttf.h>
```

Gráfico de dependência de inclusões para assets.hpp:

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Image

  Encapsula carregamento e desenho de bitmaps.
- class Music

  Gerencia reprodução em loop de faixas de áudio.
- class Sound

  Carrega e reproduz samples de efeito sonoro.
- class TextFont

  Wrapper para fontes bitmap/TTF e texto colorido.

Funções

- void loadGlobalAssets ()

  Carrega recursos globais utilizados em múltiplas cenas.
- void unloadGlobalAssets ()

  Libera os recursos globais previamente carregados.

Variáveis

- std::unique_ptr< Sound > selectSound

6.3.1 Descrição detalhada

Declara classes utilitárias para carregar e usar imagens, sons e fontes no Allegro.

Fornece wrappers RAII (`Image`, `Sound`, `Music`, `TextFont`) e helpers para assets globais utilizados em vários estados da aplicação.

6.3.2 Funções

6.3.2.1 loadGlobalAssets()

```
void loadGlobalAssets ( )
```

Carrega recursos globais utilizados em múltiplas cenas.

Atualmente apenas o efeito de som de seleção de menu é carregado, porém a função pode ser estendida para outros assets compartilhados. Esse é o diagrama das funções que utilizam essa função:

6.3.2.2 unloadGlobalAssets()

```
void unloadGlobalAssets ( )
```

Libera os recursos globais previamente carregados.

Esse é o diagrama das funções que utilizam essa função:

6.3.3 Variáveis

6.3.3.1 selectSound

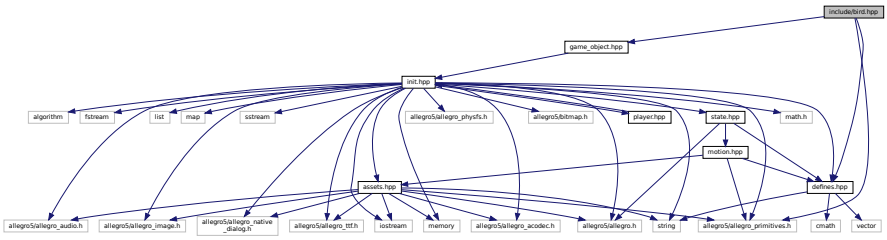
```
std::unique_ptr<Sound> selectSound  [extern]
```

6.4 Referência do Arquivo include/bird.hpp

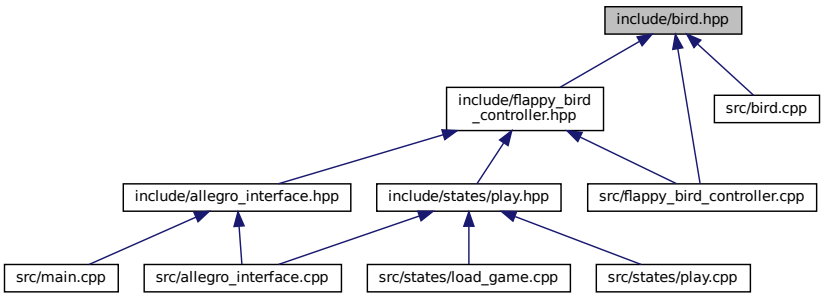
Declaracao da classe Bird (personagem controlavel).

```
#include <allegro5/allegro_primitives.h>
#include "defines.hpp"
#include "game_object.hpp"
```

Gráfico de dependência de inclusões para bird.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Bird

  Personagem controlável: gerencia movimento, animação de asas, pulo e detecção de colisões.

6.4.1 Descrição detalhada

Declaracao da classe Bird (personagem controlavel).

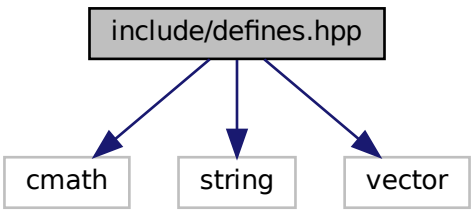
Define logica de movimento, animacao e colisao do passaro. Comentarios seguem padrao Doxygen e sem acentos.

6.5 Referência do Arquivo include/defines.hpp

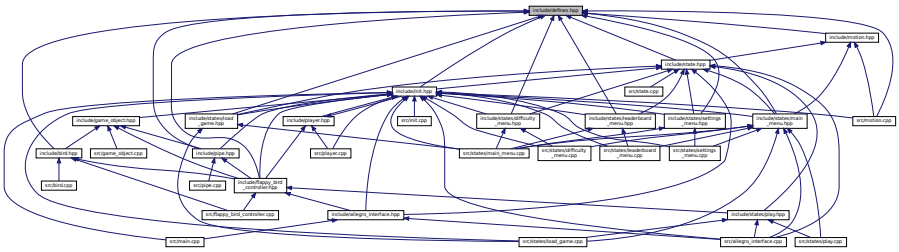
Constantes globais, macros e estruturas auxiliares do jogo.

```
#include <cmath>
#include <string>
#include <vector>
```

Gráfico de dependência de inclusões para defines.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- struct Button

  Estrutura auxiliar que representa um botão de menu e seu estado de seleção.

Definições e Macros

- #define SCREEN_H 720
- #define SCREEN_W 1280
- #define FPS 60.0
- #define MAX_INPUT_LENGTH 33
- #define JUMP_FORCE -15.0
- #define GRAVITY 1.0
- #define ROTATION 0.05
- #define X_INIT 350
- #define GAP_SIZE 250
- #define GAP_X 300
- #define BIRD_VEL 3
- #define PIPE_SPEED 5.5
- #define PIPE_VERTICAL_SPEED 1.5
- #define OSCILATION 3.0
- #define TIME_GIF_BIRD 8
- #define TETO_BIRD 0.368

Enumerações

- enum ScreenState { PLAY , PAUSE , GAME_OVER }

Variáveis

- const float MAX_ROTATION_UP = -M_PI / 6.0f
- const float MAX_ROTATION_DOWN = M_PI / 6.0f

6.5.1 Descrição detalhada

Constantes globais, macros e estruturas auxiliares do jogo.

Contém dimensões de tela, física do pássaro, velocidades, gaps dos canos e outros valores compartilhados entre módulos.

6.5.2 Definições e macros

6.5.2.1 BIRD_VEL

```
#define BIRD_VEL 3
```

6.5.2.2 FPS

```
#define FPS 60.0
```

6.5.2.3 GAP_SIZE

```
#define GAP_SIZE 250
```

6.5.2.4 GAP_X

```
#define GAP_X 300
```

6.5.2.5 GRAVITY

```
#define GRAVITY 1.0
```

6.5.2.6 JUMP_FORCE

```
#define JUMP_FORCE -15.0
```

6.5.2.7 MAX_INPUT_LENGTH

```
#define MAX_INPUT_LENGTH 33
```

6.5.2.8 OSCILATION

```
#define OSCILATION 3.0
```

6.5.2.9 PIPE_SPEED

```
#define PIPE_SPEED 5.5
```

6.5.2.10 PIPE_VERTICAL_SPEED

```
#define PIPE_VERTICAL_SPEED 1.5
```

6.5.2.11 ROTATION

```
#define ROTATION 0.05
```

6.5.2.12 SCREEN_H

```
#define SCREEN_H 720
```

6.5.2.13 SCREEN_W

```
#define SCREEN_W 1280
```

6.5.2.14 TETO_BIRD

```
#define TETO_BIRD 0.368
```

6.5.2.15 TIME_GIF_BIRD

```
#define TIME_GIF_BIRD 8
```

6.5.2.16 X_INIT

```
#define X_INIT 350
```

6.5.3 Enumerações

6.5.3.1 ScreenState

```
enum ScreenState
```

Enumeradores

PLAY	
PAUSE	
GAME_OVER	

6.5.4 Variáveis

6.5.4.1 MAX_ROTATION_DOWN

```
const float MAX_ROTATION_DOWN = M_PI / 6.0f
```

6.5.4.2 MAX_ROTATION_UP

```
const float MAX_ROTATION_UP = -M_PI / 6.0f
```

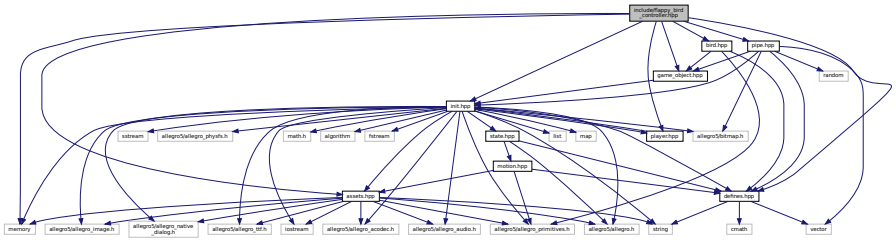
6.6 Referência do Arquivo include/flappy_bird_controller.hpp

Declaracao da classe FlappyBird (controlador principal).

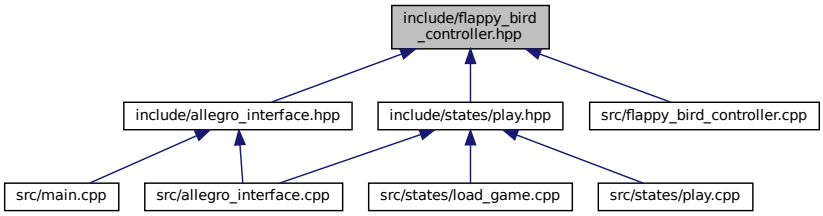
```
#include <memory>
#include "assets.hpp"
#include "bird.hpp"
#include "defines.hpp"
#include "game_object.hpp"
#include "init.hpp"
#include "pipe.hpp"
```

```
#include "player.hpp"
```

Gráfico de dependência de inclusões para flappy_bird_controller.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class FlappyBird

  Controlador principal do jogo: orquestra objetos, HUD, pontuação e fluxo de estados.

6.6.1 Descrição detalhada

Declaracao da classe FlappyBird (controlador principal).

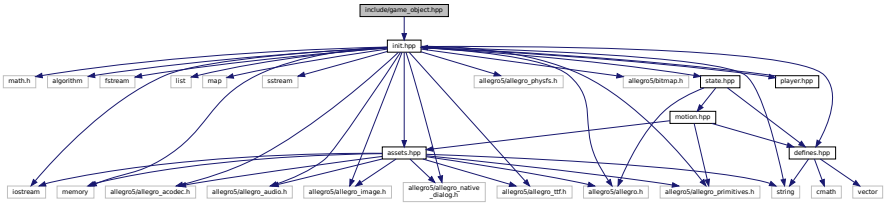
Coordena objetos, pontuacao, sons, HUD e logica de estados do jogo. Comentarios seguem padrao Doxygen sem acentos.

6.7 Referência do Arquivo include/game_object.hpp

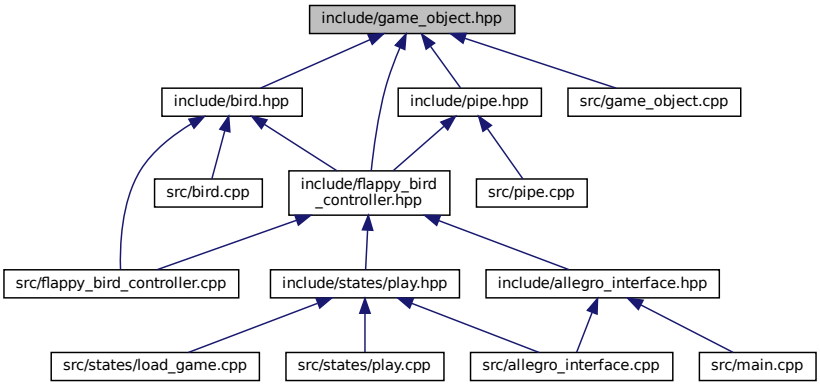
Classe base abstrata para todos os objetos renderizáveis.

```
#include "init.hpp"
```

Gráfico de dependência de inclusões para game_object.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameObject

  Objeto genérico do jogo contendo posição, sprite e hitbox.

6.7.1 Descrição detalhada

Classe base abstrata para todos os objetos renderizáveis.

Define posição, dimensões e interface para atualização/desenho. Todas as entidades que podem colidir ou ser renderizadas devem herdar desta classe e implementar os métodos puros `update()` e `draw()`.

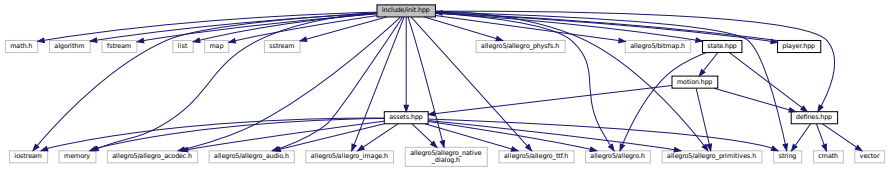
6.8 Referência do Arquivo include/init.hpp

Declara funções de inicialização e finalização do Allegro.

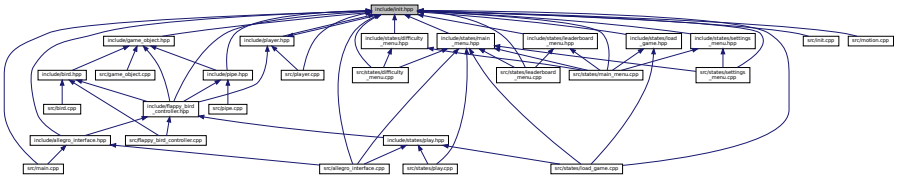
```
#include <math.h>
#include <algorithm>
#include <fstream>
#include <iostream>
#include <list>
#include <map>
#include <memory>
#include <sstream>
#include <string>
#include <allegro5/allegro.h>
#include <allegro5/allegro_acodec.h>
#include <allegro5/allegro_audio.h>
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_native_dialog.h>
#include <allegro5/allegro_physfs.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_ttf.h>
```

```
#include <allegro5/bitmap.h>
#include "assets.hpp"
#include "defines.hpp"
#include "player.hpp"
#include "state.hpp"
```

Gráfico de dependência de inclusões para init.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Funções

- void init ()

  Inicializa Allegro, janela, áudio e recursos globais.
- void deinit ()

  Libera todos os recursos e encerra o subsistema Allegro.

Variáveis

- ALLEGRO_DISPLAY ∗ display
- ALLEGRO_EVENT_QUEUE ∗ event_queue
- ALLEGRO_TIMER ∗ timer_FPS
- ALLEGRO_BITMAP ∗ icon
- ALLEGRO_KEYBOARD_STATE keystate
- Player player
- std::vector< Player > ranking
- bool g_sound_on
- int difficulty
- ALLEGRO_FONT ∗ font
- std::unique_ptr< Sound > selectSound
- std::unique_ptr< Music > backgroundMusic

6.8.1 Descrição detalhada

Declara funções de inicialização e finalização do Allegro.

Externa ponteiros globais de display, fila de eventos, timer, fontes e ranking de jogadores para serem utilizados em todo o projeto.

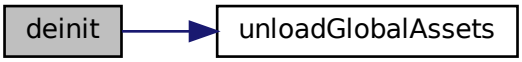
6.8.2 Funções

6.8.2.1 deinit()

```
void deinit ( )
```

Libera todos os recursos e encerra o subsistema Allegro.

Este é o diagrama das funções utilizadas por essa função:



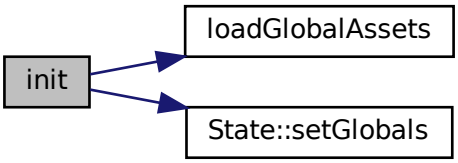
Esse é o diagrama das funções que utilizam essa função:

6.8.2.2 init()

```
void init ( )
```

Inicializa Allegro, janela, áudio e recursos globais.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

6.8.3 Variáveis

6.8.3.1 backgroundMusic

```
std::unique_ptr<Music> backgroundMusic  [extern]
```

6.8.3.2 difficulty

```
int difficulty  [extern]
```

6.8.3.3 display

```
ALLEGRO_DISPLAY* display  [extern]
```

6.8.3.4 event_queue

```
ALLEGRO_EVENT_QUEUE* event_queue  [extern]
```

6.8.3.5 font

```
ALLEGRO_FONT* font  [extern]
```

6.8.3.6 g_sound_on

```
bool g_sound_on  [extern]
```

6.8.3.7 icon

```
ALLEGRO_BITMAP* icon  [extern]
```

6.8.3.8 keystate

```
ALLEGRO_KEYBOARD_STATE keystate  [extern]
```

6.8.3.9 player

```
Player player  [extern]
```

6.8.3.10 ranking

```
std::vector<Player> ranking  [extern]
```

6.8.3.11 selectSound

```
std::unique_ptr<Sound> selectSound  [extern]
```

6.8.3.12 timer_FPS

```
ALLEGRO_TIMER* timer_FPS  [extern]
```

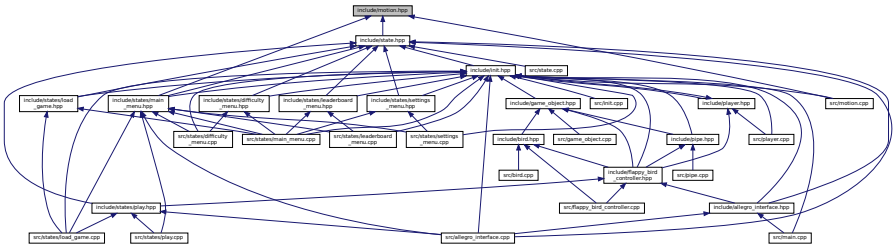
6.9 Referência do Arquivo include/motion.hpp

Gerencia efeitos de parallax e animações de cenário.

```
#include <allegro5/allegro_primitives.h>
#include "assets.hpp"
#include "defines.hpp"
```

Gráfico de dependência de inclusões para motion.hpp:

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Motion

  Responsável por atualizar e desenhar efeitos de parallax, clima e piso animado.

6.9.1 Descrição detalhada

Gerencia efeitos de parallax e animações de cenário.

Responsável por atualizar posições de fundos, partículas (chuva, neve) e elementos decorativos, possibilitando diferentes temas climáticos durante o jogo.

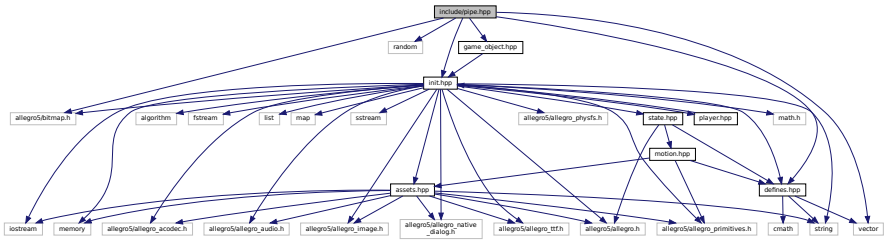
6.10 Referência do Arquivo include/pipe.hpp

Declaration of pipe-related classes (Pipe, PipePair, PipeList).

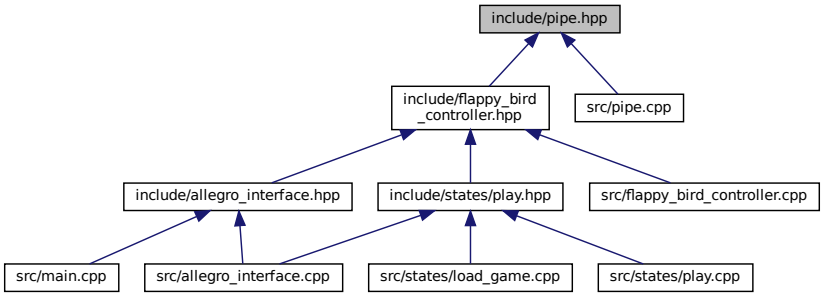
```
#include <allegro5/bitmap.h>
#include <random>
#include <vector>
#include "defines.hpp"
#include "game_object.hpp"
```

```
#include "init.hpp"
```

Gráfico de dependência de inclusões para pipe.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Pipe

  Obstáculo individual que se move horizontalmente e gera pontuação.
- class PipePair

  Agrupa dois canos (superior e inferior) podendo ter movimento vertical.
- class PipeList

  Gerencia vetor de PipePair: spawn, atualização, colisão e score.

6.10.1 Descrição detalhada

Declaration of pipe-related classes (Pipe, PipePair, PipeList).

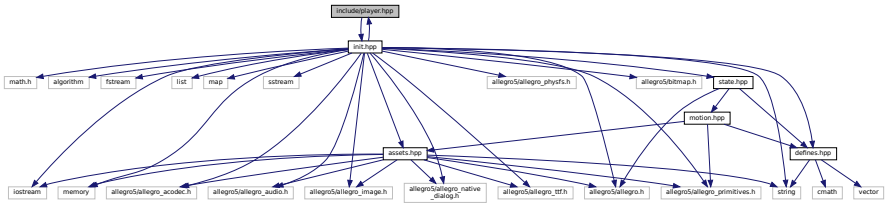
These classes implement the obstacle logic: horizontal/vertical movement, collision detection, score handling and difficulty management for the Flappy-style game.

6.11 Referência do Arquivo include/player.hpp

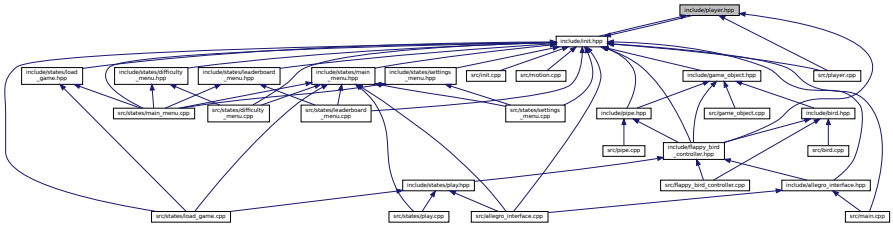
Modela um jogador e operações de ranking.

```
#include "init.hpp"
```

Gráfico de dependência de inclusões para player.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Player

  Armazena nome e pontuação e provê utilidades para ranking.

6.11.1 Descrição detalhada

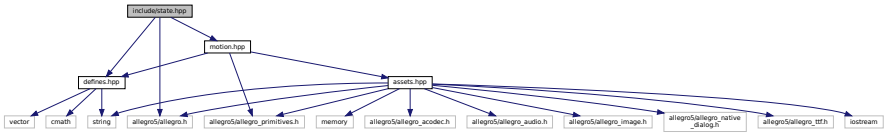
Modela um jogador e operações de ranking.

6.12 Referência do Arquivo include/state.hpp

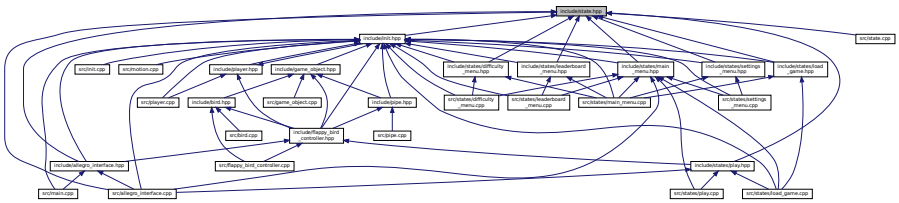
Interface base para estados da máquina de estados do jogo.

```
#include <allegro5/allegro.h>
#include "defines.hpp"
#include "motion.hpp"
```

Gráfico de dependência de inclusões para state.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class State

  Define a interface que cada tela/estado deve implementar.

6.12.1 Descrição detalhada

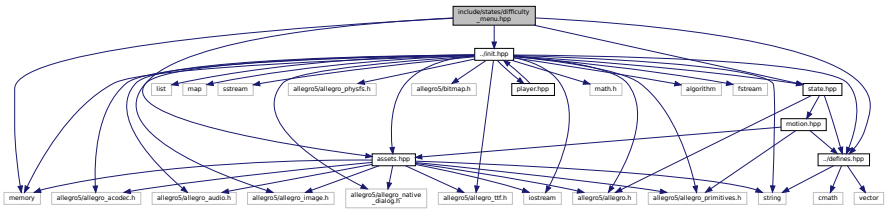
Interface base para estados da máquina de estados do jogo.

6.13 Referência do Arquivo include/states/difficulty_menu.hpp

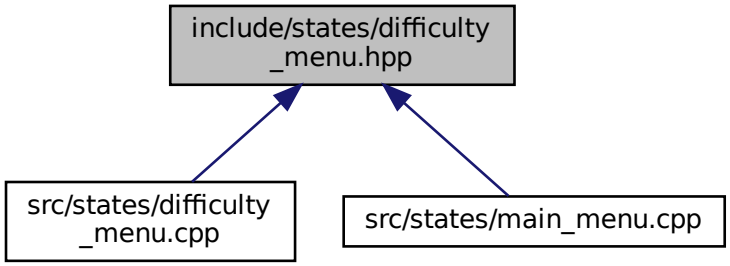
Definição da classe DifficultyMenu, que representa o estado do menu de seleção de dificuldade.

```
#include <memory>
#include "../defines.hpp"
#include "../init.hpp"
#include "../state.hpp"
#include "../assets.hpp"
```

Gráfico de dependência de inclusões para difficulty_menu.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class DifficultyMenu

  Gerencia a lógica e a renderização da tela de seleção de dificuldade.

6.13.1 Descrição detalhada

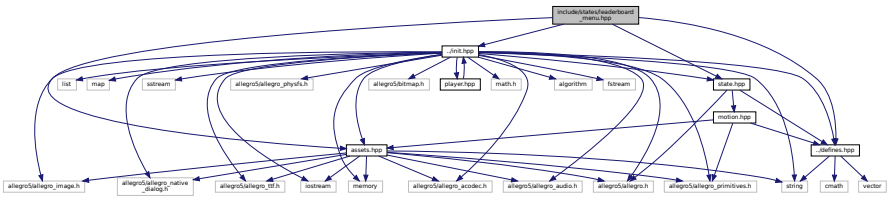
Definição da classe DifficultyMenu, que representa o estado do menu de seleção de dificuldade.

6.14 Referência do Arquivo include/states/leaderboard_menu.hpp

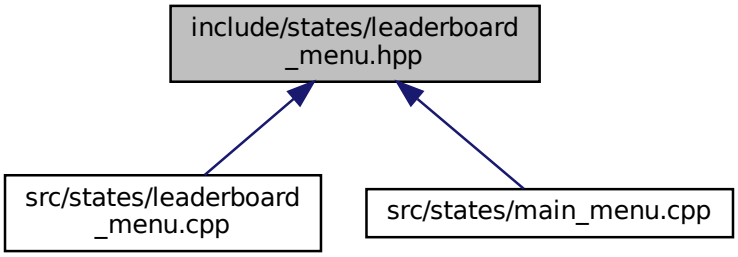
Definição da classe LeaderboardMenu, que representa o estado da tela de ranking.

```
#include "../defines.hpp"
#include "../init.hpp"
#include "../state.hpp"
#include "../assets.hpp"
```

Gráfico de dependência de inclusões para leaderboard_menu.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class LeaderboardMenu

  Exibe ranking de jogadores e permite retorno ao menu principal.

6.14.1 Descrição detalhada

Definição da classe LeaderboardMenu, que representa o estado da tela de ranking.

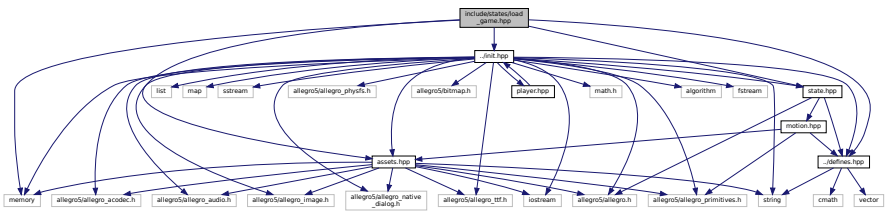
Este cabeçalho declara a interface da classe LeaderboardMenu, responsável por carregar e exibir a lista de melhores pontuações do jogo.

6.15 Referência do Arquivo include/states/load_game.hpp

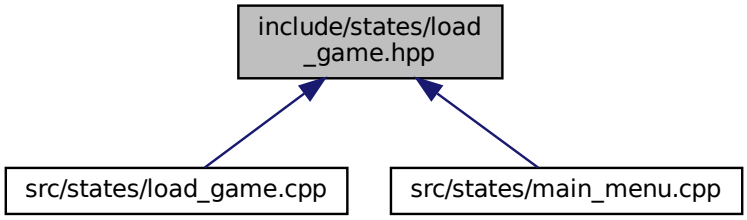
Definição da classe LoadName, que representa o estado de inserção de nome.

```
#include <memory>
#include "../defines.hpp"
#include "../init.hpp"
#include "../state.hpp"
#include "../../include/assets.hpp"
```

Gráfico de dependência de inclusões para load_game.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class LoadName

  Tela de inserção de nome usada por Novo Jogo e Carregar Jogo.

Enumerações

- enum insertNameSituations {
  noName , existName , noexistName , noError ,
  successInsert }

6.15.1 Descrição detalhada

Definição da classe LoadName, que representa o estado de inserção de nome.

Este cabeçalho declara a interface para a tela onde o jogador digita seu nome para criar um novo jogo ou carregar um jogo existente.

6.15.2 Enumerações

6.15.2.1 insertNameSituations

```
enum insertNameSituations
```

Bibliotecas necessárias

Enumeradores

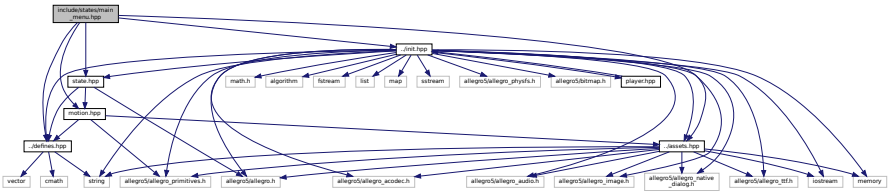
noName	
existName	
noexistName	
noError	
successInsert	

6.16 Referência do Arquivo include/states/main_menu.hpp

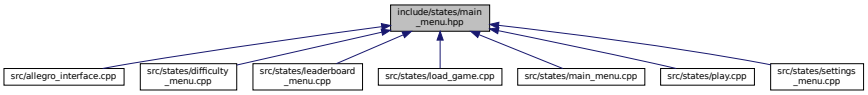
Definição da classe MainMenu, o estado que representa o menu principal do jogo.

```
#include "../assets.hpp"
#include "../defines.hpp"
#include "../init.hpp"
#include "../motion.hpp"
#include "../state.hpp"
```

Gráfico de dependência de inclusões para main_menu.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class MainMenu

  Tela inicial do jogo, centraliza navegação para demais estados.

Variáveis

- std::string inputNameScreen

6.16.1 Descrição detalhada

Definição da classe MainMenu, o estado que representa o menu principal do jogo.

Este cabeçalho declara a interface da classe MainMenu, que serve como o ponto central de navegação para todas as outras telas do jogo.

6.16.2 Variáveis

6.16.2.1 inputNameScreen

```
std::string inputNameScreen  [extern]
```

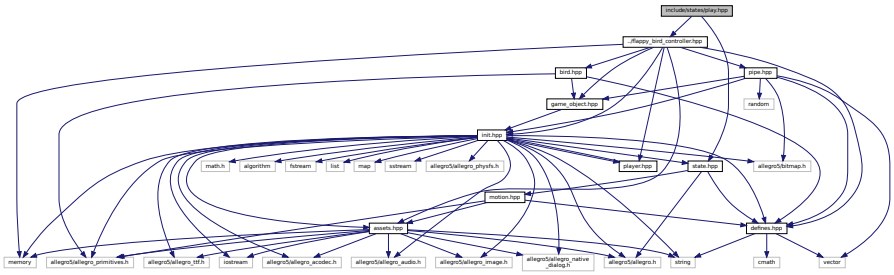
Bibliotecas necessárias

6.17 Referência do Arquivo include/states/play.hpp

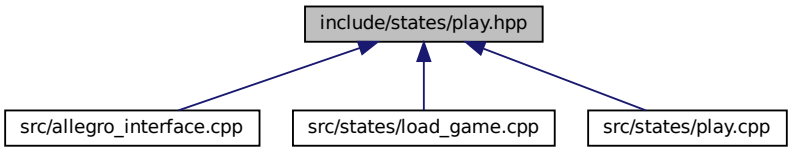
Declaracao da classe Play, responsavel pelo estado de jogo ativo.

```
#include "../flappy_bird_controller.hpp"
#include "../state.hpp"
```

Gráfico de dependência de inclusões para play.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Play

  Estado principal de jogo: controla loop ativo, pontuação e game over.

6.17.1 Descrição detalhada

Declaracao da classe Play, responsavel pelo estado de jogo ativo.

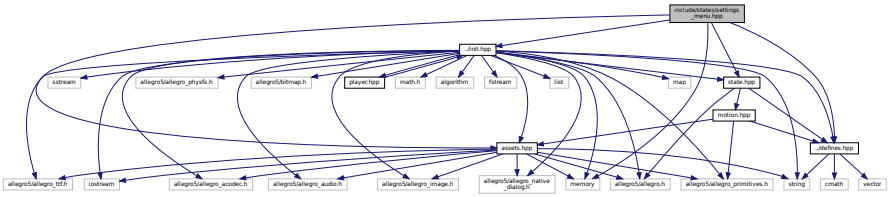
Este cabecalho define a classe Play, que orquestra a jogatina, incluindo controle de entrada, atualizacao de logica e renderizacao.

6.18 Referência do Arquivo include/states/settings_menu.hpp

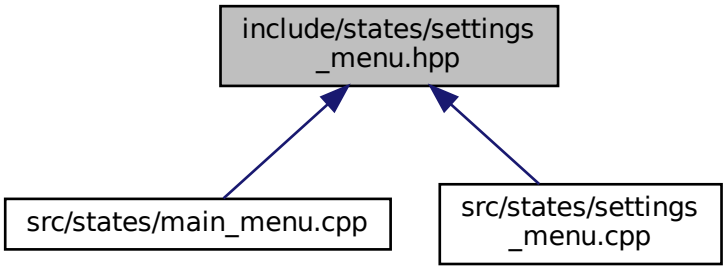
Definição da classe SettingsMenu, que representa o estado do menu de configurações.

```
#include "../defines.hpp"
#include "../init.hpp"
#include "../state.hpp"
#include "../assets.hpp"
#include <memory>
```

Gráfico de dependência de inclusões para settings_menu.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class SettingsMenu

  Tela de configurações: permite alterar clima e habilitar/desabilitar música.

6.18.1 Descrição detalhada

Definição da classe SettingsMenu, que representa o estado do menu de configurações.

Este cabeçalho declara a interface da classe SettingsMenu, responsável por permitir ao jogador ajustar opções como música e clima.

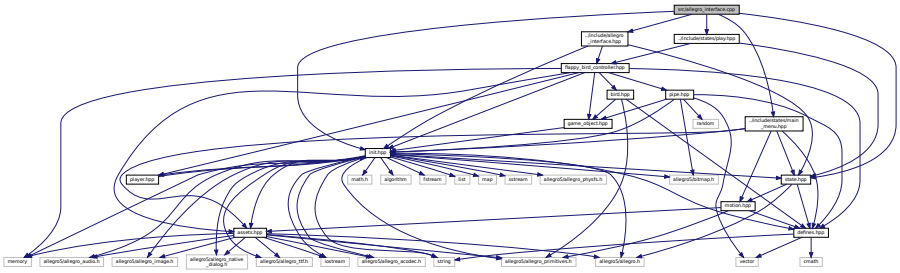
6.19 Referência do Arquivo README.md

6.20 Referência do Arquivo src/allegro_interface.cpp

Implementação do controlador principal que integra Allegro às lógicas de estado do jogo.

```
#include "../include/allegro_interface.hpp"
#include "../include/init.hpp"
#include "../include/state.hpp"
#include "../include/states/main_menu.hpp"
#include "../include/states/play.hpp"
```

Gráfico de dependência de inclusões para allegro_interface.cpp:



6.20.1 Descrição detalhada

Implementação do controlador principal que integra Allegro às lógicas de estado do jogo.

Este arquivo contém a implementação do `AllegroController`, responsável por inicializar o primeiro estado, executar o loop de eventos da biblioteca Allegro e orquestrar transições de estado, atualizações e renderizações.

6.21 Referência do Arquivo src/assets.cpp

Carregamento e gerenciamento de assets gráficos/áudio.

```
#include "../include/assets.hpp"
#include <memory>
```

Gráfico de dependência de inclusões para assets.cpp:

Funções

- void loadGlobalAssets ()

  Carrega recursos globais utilizados em múltiplas cenas.
- void unloadGlobalAssets ()

  Libera os recursos globais previamente carregados.

Variáveis

- std::unique_ptr< Sound > selectSound

6.21.1 Descrição detalhada

Carregamento e gerenciamento de assets gráficos/áudio.

6.21.2 Funções

6.21.2.1 loadGlobalAssets()

```
void loadGlobalAssets ( )
```

Carrega recursos globais utilizados em múltiplas cenas.

Atualmente apenas o efeito de som de seleção de menu é carregado, porém a função pode ser estendida para outros assets compartilhados. Esse é o diagrama das funções que utilizam essa função:

6.21.2.2 unloadGlobalAssets()

```
void unloadGlobalAssets ( )
```

Libera os recursos globais previamente carregados.

Esse é o diagrama das funções que utilizam essa função:

6.21.3 Variáveis

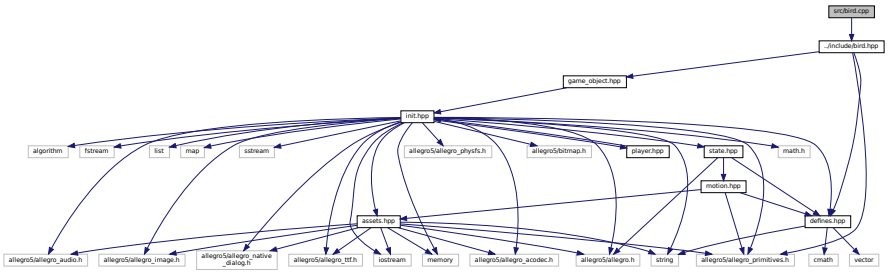
6.21.3.1 selectSound

```
std::unique_ptr<Sound> selectSound
```

6.22 Referência do Arquivo src/bird.cpp

Implementação da classe Bird (animação, física e colisões).

```
#include "../include/bird.hpp"
```
Gráfico de dependência de inclusões para bird.cpp:



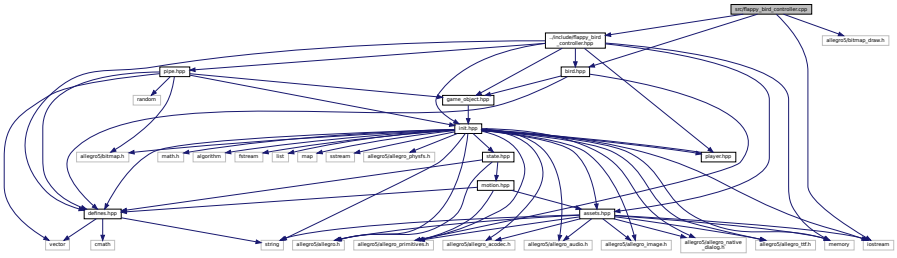
6.22.1 Descrição detalhada

Implementação da classe Bird (animação, física e colisões).

6.23 Referência do Arquivo src/flappy_bird_controller.cpp

Controlador de fluxo do jogo (pontuação, eventos globais).

```
#include "../include/flappy_bird_controller.hpp"
#include <allegro5/bitmap_draw.h>
#include <iostream>
#include "../include/bird.hpp"
```
Gráfico de dependência de inclusões para flappy_bird_controller.cpp:



6.23.1 Descrição detalhada

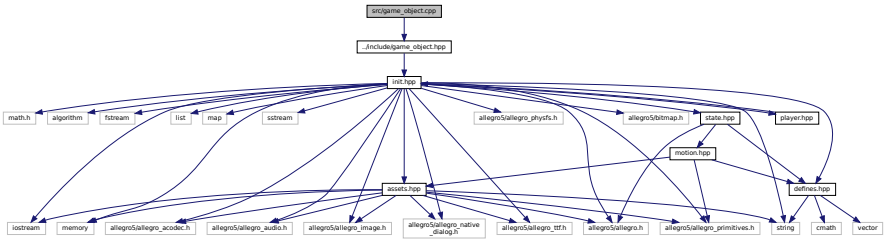
Controlador de fluxo do jogo (pontuação, eventos globais).

6.24 Referência do Arquivo src/game_object.cpp

Métodos utilitários da classe base GameObject.

```
#include "../include/game_object.hpp"
```

Gráfico de dependência de inclusões para game_object.cpp:



6.24.1 Descrição detalhada

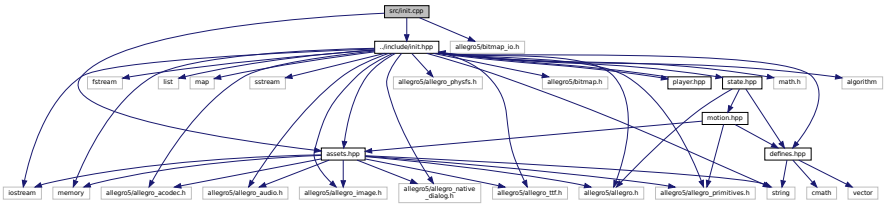
Métodos utilitários da classe base GameObject.

6.25 Referência do Arquivo src/init.cpp

Inicialização de Allegro, fontes, addons e recursos globais.

```
#include "../include/init.hpp"
#include <allegro5/bitmap_io.h>
#include "../include/assets.hpp"
```

Gráfico de dependência de inclusões para init.cpp:



Definições e Macros

- #define LOG(x) std::cerr << x << '\n'

Funções

- void init ()

  Inicializa Allegro, janela, áudio e recursos globais.
- void deinit ()

  Libera todos os recursos e encerra o subsistema Allegro.

Variáveis

- ALLEGRO_DISPLAY * display = nullptr
- ALLEGRO_EVENT_QUEUE * event_queue = nullptr
- ALLEGRO_TIMER * timer_FPS = nullptr
- Player player = Player(" ", 0)
- std::vector< Player > ranking = player.ReadLeaderboard("Leaderboard.txt")
- bool g_sound_on = true
- int difficulty = 1
- ALLEGRO_FONT * font = NULL
- ALLEGRO_BITMAP * icon = NULL
- std::unique_ptr< Music > backgroundMusic

6.25.1 Descrição detalhada

Inicialização de Allegro, fontes, addons e recursos globais.

6.25.2 Definições e macros

6.25.2.1 LOG

```
#define LOG(
             x ) std::cerr << x << '\n'
```

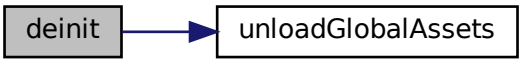
6.25.3 Funções

6.25.3.1 deinit()

```
void deinit ( )
```

Libera todos os recursos e encerra o subsistema Allegro.

Este é o diagrama das funções utilizadas por essa função:



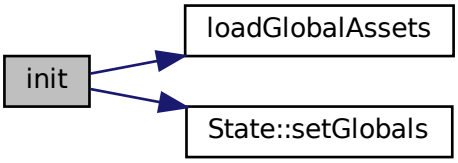
Esse é o diagrama das funções que utilizam essa função:

6.25.3.2 init()

```
void init ( )
```

Inicializa Allegro, janela, áudio e recursos globais.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

6.25.4 Variáveis

6.25.4.1 backgroundMusic

```
std::unique_ptr<Music> backgroundMusic
```

6.25.4.2 difficulty

```
int difficulty = 1
```

6.25.4.3 display

```
ALLEGRO_DISPLAY* display = nullptr
```

6.25.4.4 event_queue

```
ALLEGRO_EVENT_QUEUE* event_queue = nullptr
```

6.25.4.5 font

```
ALLEGRO_FONT* font = NULL
```

6.25.4.6 g_sound_on

```
bool g_sound_on = true
```

6.25.4.7 icon

```
ALLEGRO_BITMAP* icon = NULL
```

6.25.4.8 player

```
Player player = Player(" ", 0)
```

6.25.4.9 ranking

```
std::vector<Player> ranking = player.ReadLeaderboard("Leaderboard.txt")
```

6.25.4.10 timer_FPS

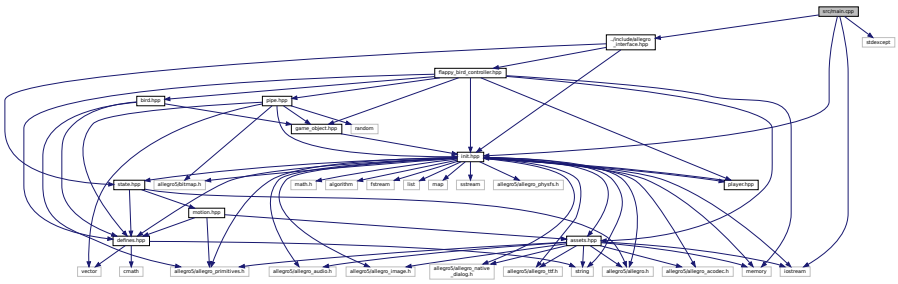
```
ALLEGRO_TIMER* timer_FPS = nullptr
```

6.26 Referência do Arquivo src/main.cpp

Ponto de entrada do jogo Flappy Bird.

```
#include "../include/allegro_interface.hpp"
#include "../include/init.hpp"
#include <iostream>
#include <stdexcept>
```

Gráfico de dependência de inclusões para main.cpp:



Funções

- int main ()

  Função principal que inicializa, executa e encerra o jogo.

6.26.1 Descrição detalhada

Ponto de entrada do jogo Flappy Bird.

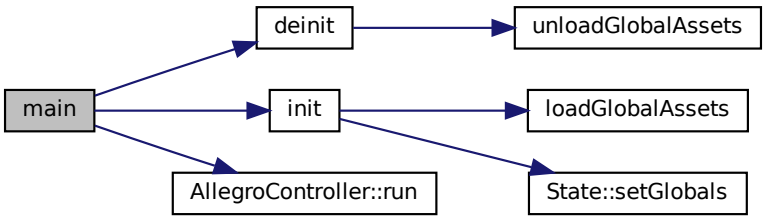
6.26.2 Funções

6.26.2.1 main()

```
int main ( )
```

Função principal que inicializa, executa e encerra o jogo.

Este é o diagrama das funções utilizadas por essa função:

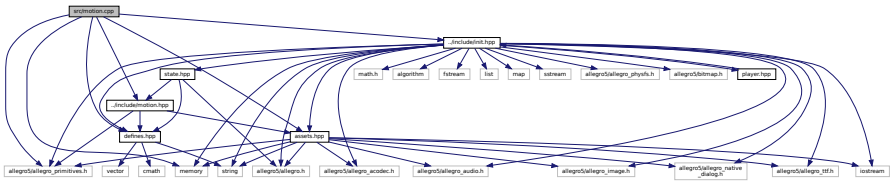


6.27 Referência do Arquivo src/motion.cpp

Implementação da classe Motion (parallax e clima).

```
#include "../include/motion.hpp"
#include <allegro5/allegro_primitives.h>
#include <memory>
#include "../include/defines.hpp"
#include "../include/init.hpp"
#include "assets.hpp"
```

Gráfico de dependência de inclusões para motion.cpp:



6.27.1 Descrição detalhada

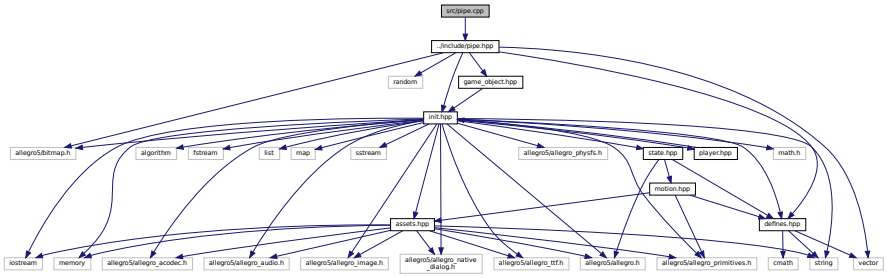
Implementação da classe Motion (parallax e clima).

6.28 Referência do Arquivo src/pipe.cpp

Implementação das classes Pipe, PipePair e PipeList.

```
#include "../include/pipe.hpp"
```

Gráfico de dependência de inclusões para pipe.cpp:



6.28.1 Descrição detalhada

Implementação das classes Pipe, PipePair e PipeList.

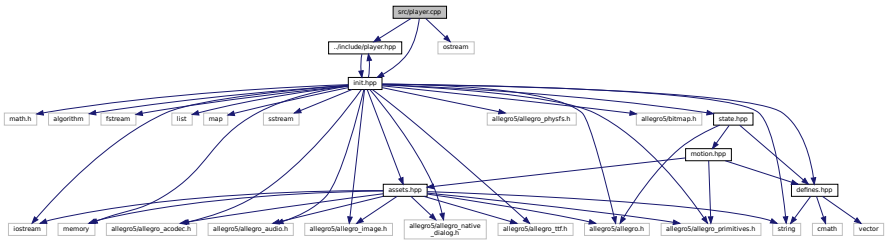
Contém a lógica de movimento, verificação de colisão e contagem de pontos do sistema de obstáculos (canos).

6.29 Referência do Arquivo src/player.cpp

Implementação da classe Player e utilidades de ranking.

```
#include "../include/player.hpp"
#include <ostream>
#include "../include/init.hpp"
```

Gráfico de dependência de inclusões para player.cpp:



Funções

- std::ostream & operator<< (std::ostream &os, const Player &p)
  Sobrecarga de operador de streaming para debug.

6.29.1 Descrição detalhada

Implementação da classe Player e utilidades de ranking.

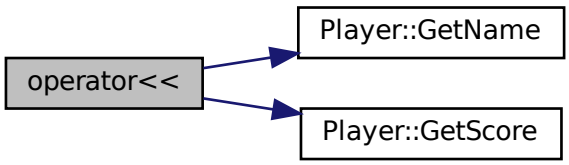
6.29.2 Funções

6.29.2.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Player & p )
```

Sobrecarga de operador de streaming para debug.

Este é o diagrama das funções utilizadas por essa função:

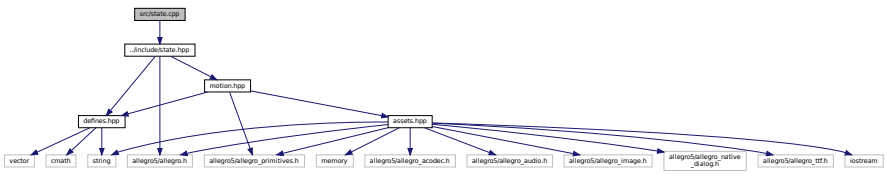


6.30 Referência do Arquivo src/state.cpp

Implementação base de State para máquina de estados do jogo.

```
#include "../include/state.hpp"
```

Gráfico de dependência de inclusões para state.cpp:



6.30.1 Descrição detalhada

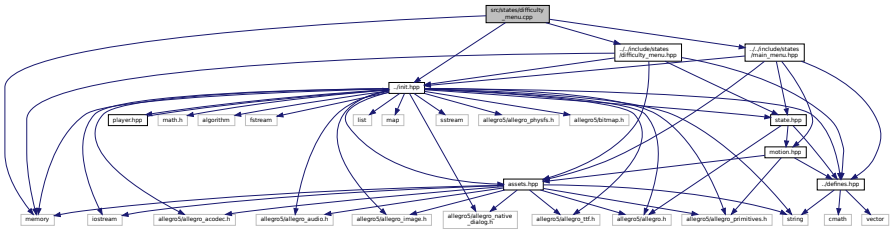
Implementação base de State para máquina de estados do jogo.

6.31 Referência do Arquivo src/states/difficulty_menu.cpp

Implementação da classe DifficultyMenu, que gerencia a tela de seleção de dificuldade.

```
#include <memory>
#include "../../include/states/difficulty_menu.hpp"
#include "../../include/init.hpp"
#include "../../include/states/main_menu.hpp"
```

Gráfico de dependência de inclusões para difficulty_menu.cpp:



6.31.1 Descrição detalhada

Implementação da classe DifficultyMenu, que gerencia a tela de seleção de dificuldade.

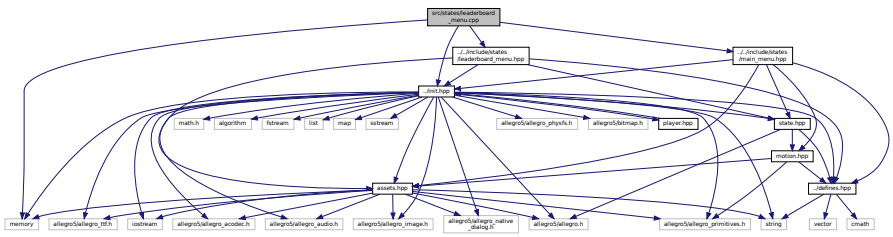
Este módulo contém a lógica para manipulação de entrada do usuário, atualização de estado e renderização dos elementos gráficos da tela de dificuldade.

6.32 Referência do Arquivo src/states/leaderboard_menu.cpp

Implementação da classe LeaderboardMenu, responsável por exibir a tela de ranking.

```
#include "../../include/states/leaderboard_menu.hpp"
#include "../../include/init.hpp"
#include "../../include/states/main_menu.hpp"
#include <memory>
```

Gráfico de dependência de inclusões para leaderboard_menu.cpp:



6.32.1 Descrição detalhada

Implementação da classe LeaderboardMenu, responsável por exibir a tela de ranking.

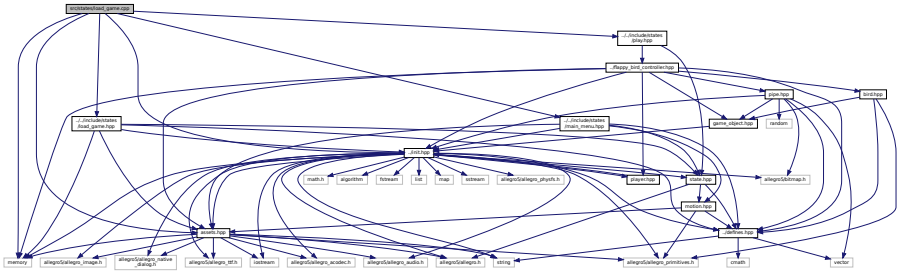
Este módulo carrega, ordena e exibe a pontuação dos jogadores, além de gerenciar a interação do usuário nesta tela.

6.33 Referência do Arquivo src/states/load_game.cpp

Implementação da classe LoadName, responsável pela tela de inserção de nome do jogador.

```
#include "../../include/states/load_game.hpp"
#include <memory>
#include "../../include/assets.hpp"
#include "../../include/init.hpp"
#include "../../include/states/main_menu.hpp"
#include "../../include/states/play.hpp"
```

Gráfico de dependência de inclusões para load_game.cpp:



6.33.1 Descrição detalhada

Implementação da classe LoadName, responsável pela tela de inserção de nome do jogador.

Este módulo gerencia a entrada de texto para o nome do jogador, valida o nome para um novo jogo ou carrega um jogador existente, e lida com a navegação e exibição de erros.

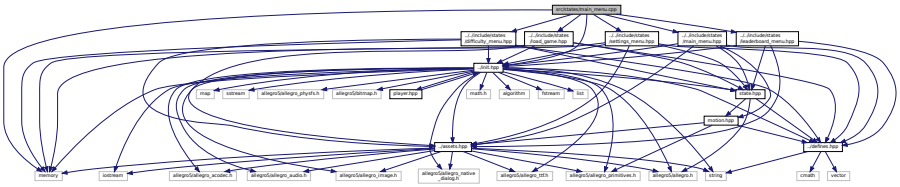
6.34 Referência do Arquivo src/states/main_menu.cpp

Implementação da classe MainMenu, o estado principal e tela inicial do jogo.

```
#include "../../include/states/main_menu.hpp"
#include "../../include/init.hpp"
#include "../../include/states/difficulty_menu.hpp"
#include "../../include/states/leaderboard_menu.hpp"
#include "../../include/states/load_game.hpp"
#include "../../include/states/settings_menu.hpp"
```

```
#include <memory>
```

Gráfico de dependência de inclusões para main_menu.cpp:



Variáveis

- std::string inputNameScreen = ""

6.34.1 Descrição detalhada

Implementação da classe MainMenu, o estado principal e tela inicial do jogo.

Este módulo gerencia a navegação do menu principal, o carregamento de todos os recursos visuais da tela e a transição para outros estados do jogo (Novo Jogo, Carregar, Configurações, etc.).

6.34.2 Variáveis

6.34.2.1 inputNameScreen

```
std::string inputNameScreen = ""
```

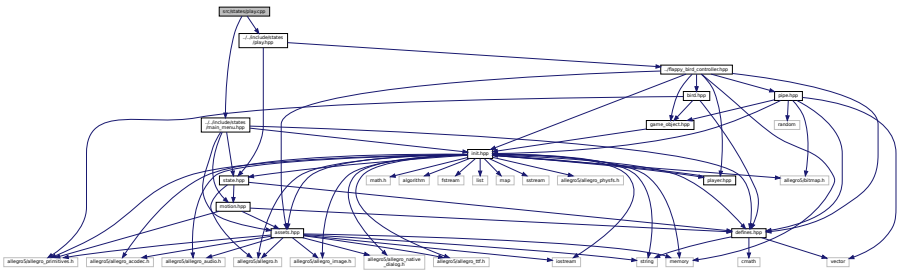
Bibliotecas necessárias

6.35 Referência do Arquivo src/states/play.cpp

Implementacao da classe Play.

```
#include "../../include/states/play.hpp"
#include "../../include/states/main_menu.hpp"
```

Gráfico de dependência de inclusões para play.cpp:



6.35.1 Descrição detalhada

Implementacao da classe Play.

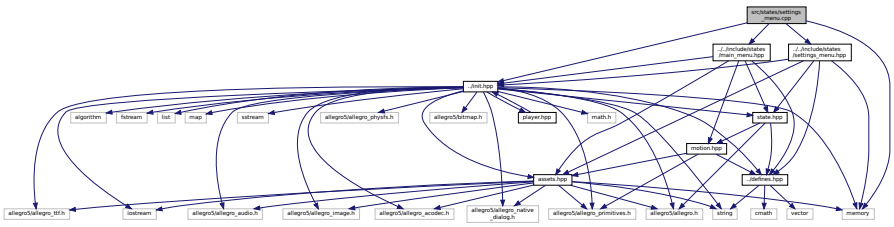
Controla o loop principal de jogo: entrada, atualizacao e desenho, alem de transicoes para pausa e game over.

6.36 Referência do Arquivo src/states/settings_menu.cpp

Implementação da classe SettingsMenu, que gerencia a tela de configurações do jogo.

```
#include "../../include/states/settings_menu.hpp"
#include <memory>
#include "../../include/init.hpp"
#include "../../include/states/main_menu.hpp"
```

Gráfico de dependência de inclusões para settings_menu.cpp:



6.36.1 Descrição detalhada

Implementação da classe SettingsMenu, que gerencia a tela de configurações do jogo.

Este módulo contém a lógica para manipulação de entrada, atualização de estado e renderização dos elementos gráficos da tela de configurações.

Índice Remissivo

~AllegroController
  AllegroController, 10
~Bird
  Bird, 15
~GameObject
  GameObject, 45
~Image
  Image, 52
~Music
  Music, 79
~Sound
  Sound, 115
~State
  State, 116
~TextFont
  TextFont, 119

add_pipe_pair
  PipeList, 86
AllegroController, 9
  ~AllegroController, 10
  AllegroController, 10
  cleanup, 10
  current_state, 12
  game, 12
  initialize, 11
  load, 11
  motion, 12
  run, 11
  screen_height, 12
  screen_width, 12
assets.cpp
  loadGlobalAssets, 149
  selectSound, 150
  unloadGlobalAssets, 149
assets.hpp
  loadGlobalAssets, 125
  selectSound, 126
  unloadGlobalAssets, 125

b
  TextFont, 120
background
  Motion, 73
background_rain
  Motion, 73
background_snow
  Motion, 73
backgroundMusic
  init.cpp, 154
  init.hpp, 136
Bird, 13
  ~Bird, 15
  Bird, 14
  breakanimation, 20
  check_bird_collision, 15
  contB, 20
  destroy_bitmaps, 16
  draw, 16
  frame1, 21
  frame2, 21
  frame3, 21
  frames, 21
  gravity, 21
  jump, 17
  jumpForce, 21
  loop_animation, 17
  reset_xy, 18
  rotation, 22
  set_break, 18
  set_by, 19
  update, 19
  valueB, 22
  vy, 22
bird1
  FlappyBird, 40
  Motion, 73
bird2
  FlappyBird, 40
  Motion, 73
bird3
  FlappyBird, 40
  Motion, 73
BIRD_VEL
  defines.hpp, 129
bottom
  PipePair, 94
breakanimation
  Bird, 20
breaker
  FlappyBird, 31
Button, 22
  buttonSelectState, 23
  name, 23
buttonBackDeselect
  LoadName, 62
  SettingsMenu, 111
buttonBackSelect
  LoadName, 62

SettingsMenu, 111
buttonDifficultyDeselect
MainMenu, 67
buttonDifficultyEasy
DifficultyMenu, 27
buttonDifficultyHard
DifficultyMenu, 27
buttonDifficultyNormal
DifficultyMenu, 27
buttonDifficultySelect
MainMenu, 67
buttonExitDeselect
MainMenu, 67
Play, 99
buttonExitSelect
MainMenu, 68
Play, 99
buttonInsertDeselect
LoadName, 62
buttonInsertSelect
LoadName, 62
buttonLeaderboardDeselect
MainMenu, 68
buttonLeaderboardSelect
MainMenu, 68
buttonLoadGameDeselect
MainMenu, 68
buttonLoadGameSelect
MainMenu, 68
buttonMusicDaySelect
SettingsMenu, 112
buttonMusicRainSelect
SettingsMenu, 112
buttonMusicSelect
SettingsMenu, 112
buttonMusicSnowSelect
SettingsMenu, 112
buttonNewGameDeselect
MainMenu, 68
buttonNewGameSelect
MainMenu, 68
buttonNoMusicDaySelect
SettingsMenu, 112
buttonNoMusicRainSelect
SettingsMenu, 112
buttonNoMusicSnowSelect
SettingsMenu, 112
buttonPause
Play, 99
buttonPositionSelected
DifficultyMenu, 27
LoadName, 62
MainMenu, 68
Play, 100
SettingsMenu, 112
buttonSaveDeselect
DifficultyMenu, 28
buttonSaveSelect
DifficultyMenu, 28
buttonSelectState
Button, 23
buttonSettingsDeselect
MainMenu, 69
buttonSettingsSelect
MainMenu, 69
buttonTryagainDeselect
Play, 100
buttonTryagainSelect
Play, 100
by
GameObject, 49

campLeaderboard
LeaderboardMenu, 57
campSettingsMusic
SettingsMenu, 113
campSettingsNoMusic
SettingsMenu, 113
change_vel
FlappyBird, 40
change_velocity
FlappyBird, 31
check_bird_collision
Bird, 15
GameObject, 45
check_collision
PipeList, 86
check_collision_with_boundaries
GameObject, 45
check_collisions
FlappyBird, 32
check_score
Pipe, 82
PipeList, 87
CheckingName
Player, 102
cleanup
AllegroController, 10
clouds
Motion, 73
clouds2
Motion, 73
cont
Motion, 74
contB
Bird, 20
Motion, 74
contF
Motion, 74
control_pipes
FlappyBird, 32
controll
Motion, 74
controller
Motion, 74
current_state
AllegroController, 12

currentPlayer
FlappyBird, 40

defines.hpp
BIRD_VEL, 129
FPS, 129
GAME_OVER, 131
GAP_SIZE, 129
GAP_X, 129
GRAVITY, 129
JUMP_FORCE, 129
MAX_INPUT_LENGTH, 129
MAX_ROTATION_DOWN, 131
MAX_ROTATION_UP, 131
OSCILATION, 129
PAUSE, 131
PIPE_SPEED, 130
PIPE_VERTICAL_SPEED, 130
PLAY, 131
ROTATION, 130
SCREEN_H, 130
SCREEN_W, 130
ScreenState, 131
TETO_BIRD, 130
TIME_GIF_BIRD, 130
X_INIT, 130
deinit
init.cpp, 152
init.hpp, 135
delete_pipe_pair
PipeList, 88
destroy_bitmaps
Bird, 16
difficulty
init.cpp, 154
init.hpp, 136
difficulty_game
FlappyBird, 41
difficulty_pipe
PipeList, 91
DifficultyMenu, 23
buttonDifficultyEasy, 27
buttonDifficultyHard, 27
buttonDifficultyNormal, 27
buttonPositionSelected, 27
buttonSaveDeselect, 28
buttonSaveSelect, 28
DifficultyMenu, 25
difficultySelected, 28
draw, 25
enter, 26
font, 28
handle_input, 26
menuButtons, 28
update, 26
difficultySelected
DifficultyMenu, 28
display
init.cpp, 154
init.hpp, 136
State, 118
docs/mainpage.dox, 123
Draw
Image, 52
draw
Bird, 16
DifficultyMenu, 25
FlappyBird, 33
GameObject, 46
LeaderboardMenu, 55
LoadName, 59
MainMenu, 65
Motion, 71
Pipe, 83
PipeList, 88
Play, 97
SettingsMenu, 109
State, 117
draw_animated_ground
FlappyBird, 34
draw_HUD
FlappyBird, 34
draw_intial_text
FlappyBird, 35
drips1
Motion, 74
drips2
Motion, 74

enter
DifficultyMenu, 26
LeaderboardMenu, 55
LoadName, 60
MainMenu, 66
Play, 97
SettingsMenu, 110
State, 117
errorFont
LoadName, 62
errorSituation
LoadName, 62
ev
State, 118
event_queue
init.cpp, 154
init.hpp, 136
existName
load_game.hpp, 144

flakesBig
Motion, 74
flakesLittle
Motion, 75
flakesLittle2
Motion, 75
flappy
Play, 100
flappy_obj

FlappyBird, 41
FlappyBird, 29
bird1, 40
bird2, 40
bird3, 40
breaker, 31
change_vel, 40
change_velocity, 31
check_collisions, 32
control_pipes, 32
currentPlayer, 40
difficulty_game, 41
draw, 33
draw_animated_ground, 34
draw_HUD, 34
draw_intial_text, 35
flappy_obj, 41
FlappyBird, 31
gameOverSound, 41
get_bird, 35
get_pipes, 35
get_state, 35
ground, 41
ground2, 41
jump, 36
pipe, 41
pipelist, 42
pointSound, 42
positionF2_x, 42
positionF_x, 42
reset, 36
saveCurrentPlayerScore, 36
score, 42
set_current_player, 37
set_playerscore, 37
starter, 37
state, 42
time, 42
unbreaker, 38
update, 38
update_score, 39
velocity, 43
velocity_backup, 43
font
DifficultyMenu, 28
init.cpp, 154
init.hpp, 136
LeaderboardMenu, 57
Play, 100
SettingsMenu, 113
TextFont, 120
FPS
defines.hpp, 129
frame1
Bird, 21
frame2
Bird, 21
frame3
Bird, 21
frames
Bird, 21

g
TextFont, 120
g_sound_on
init.cpp, 154
init.hpp, 136
game
AllegroController, 12
GAME_OVER
defines.hpp, 131
GameObject, 43
∼GameObject, 45
by, 49
check_bird_collision, 45
check_collision_with_boundaries, 45
draw, 46
GameObject, 44
get_height, 46
get_width, 46
get_x, 47
get_x_final, 47
get_y, 47
height, 49
obj_sprite, 49
set_finals, 48
set_x, 48
set_y, 48
update, 49
width, 49
x, 49
x_final, 50
y, 50
y_final, 50
gameOverSound
FlappyBird, 41
GAP_SIZE
defines.hpp, 129
GAP_X
defines.hpp, 129
gen
PipeList, 91
get_bird
FlappyBird, 35
get_height
GameObject, 46
get_pipe_pairs
PipeList, 88
get_pipes
FlappyBird, 35
get_points
PipeList, 89
get_state
FlappyBird, 35
get_width
GameObject, 46
get_x

GameObject, 47
get_x_final
GameObject, 47
get_y
GameObject, 47
Pipe, 83
getBitmap
Image, 52
GetName
Player, 103
GetScore
Player, 103
GRAVITY
defines.hpp, 129
gravity
Bird, 21
ground
FlappyBird, 41
Motion, 75
ground2
FlappyBird, 41
Motion, 75

handle_input
DifficultyMenu, 26
LeaderboardMenu, 56
LoadName, 60
MainMenu, 66
Play, 98
SettingsMenu, 110
State, 117
height
GameObject, 49

icon
init.cpp, 155
init.hpp, 137
Image, 50
∼Image, 52
Draw, 52
getBitmap, 52
Image, 51
image, 52
x, 53
y, 53
image
Image, 52
include/allegro_interface.hpp, 123
include/assets.hpp, 124
include/bird.hpp, 126
include/defines.hpp, 127
include/flappy_bird_controller.hpp, 131
include/game_object.hpp, 132
include/init.hpp, 133
include/motion.hpp, 138
include/pipe.hpp, 138
include/player.hpp, 140
include/state.hpp, 140
include/states/difficulty_menu.hpp, 141
include/states/leaderboard_menu.hpp, 142
include/states/load_game.hpp, 143
include/states/main_menu.hpp, 145
include/states/play.hpp, 146
include/states/settings_menu.hpp, 147
init
init.cpp, 153
init.hpp, 135
init.cpp
backgroundMusic, 154
deinit, 152
difficulty, 154
display, 154
event_queue, 154
font, 154
g_sound_on, 154
icon, 155
init, 153
LOG, 152
player, 155
ranking, 155
timer_FPS, 155
init.hpp
backgroundMusic, 136
deinit, 135
difficulty, 136
display, 136
event_queue, 136
font, 136
g_sound_on, 136
icon, 137
init, 135
keystate, 137
player, 137
ranking, 137
selectSound, 137
timer_FPS, 137
initialize
AllegroController, 11
inputNameScreen
main_menu.cpp, 161
main_menu.hpp, 145
insertNameSituations
load_game.hpp, 144
is_off_screen
Pipe, 83

jump
Bird, 17
FlappyBird, 36
JUMP_FORCE
defines.hpp, 129
jumpForce
Bird, 21

keystate
init.hpp, 137

LeaderboardMenu, 53

campLeaderboard, 57
draw, 55
enter, 55
font, 57
handle_input, 56
LeaderboardMenu, 54
update, 56
lights
Motion, 75
little
Motion, 75
little2
Motion, 75
little3
Motion, 75
load
AllegroController, 11
load_game.hpp
existName, 144
insertNameSituations, 144
noError, 144
noexistName, 144
noName, 144
successInsert, 144
loadGlobalAssets
assets.cpp, 149
assets.hpp, 125
LoadName, 58
buttonBackDeselect, 62
buttonBackSelect, 62
buttonInsertDeselect, 62
buttonInsertSelect, 62
buttonPositionSelected, 62
draw, 59
enter, 60
errorFont, 62
errorSituation, 62
handle_input, 60
LoadName, 59
menuButtons, 62
nameCampDeselect, 63
nameCampSelect, 63
nameFont, 63
playerNameString, 63
update, 61
LOG
init.cpp, 152
logoGameOver
Play, 100
logoNormal
MainMenu, 69
loop_animation
Bird, 17

main
main.cpp, 156
main.cpp
main, 156
main_menu.cpp
inputNameScreen, 161
main_menu.hpp
inputNameScreen, 145
MainMenu, 64
buttonDifficultyDeselect, 67
buttonDifficultySelect, 67
buttonExitDeselect, 67
buttonExitSelect, 68
buttonLeaderboardDeselect, 68
buttonLeaderboardSelect, 68
buttonLoadGameDeselect, 68
buttonLoadGameSelect, 68
buttonNewGameDeselect, 68
buttonNewGameSelect, 68
buttonPositionSelected, 68
buttonSettingsDeselect, 69
buttonSettingsSelect, 69
draw, 65
enter, 66
handle_input, 66
logoNormal, 69
MainMenu, 65
menuButtons, 69
update, 67
MAX_INPUT_LENGTH
defines.hpp, 129
MAX_ROTATION_DOWN
defines.hpp, 131
MAX_ROTATION_UP
defines.hpp, 131
menuButtons
DifficultyMenu, 28
LoadName, 62
MainMenu, 69
Play, 100
SettingsMenu, 113
Motion, 69
background, 73
background_rain, 73
background_snow, 73
bird1, 73
bird2, 73
bird3, 73
clouds, 73
clouds2, 73
cont, 74
contB, 74
contF, 74
controll, 74
controller, 74
draw, 71
drips1, 74
drips2, 74
flakesBig, 74
flakesLittle, 75
flakesLittle2, 75
ground, 75
ground2, 75

lights, 75
little, 75
little2, 75
little3, 75
Motion, 71
positionB_x, 76
positionC_x, 76
positionCC_x, 76
positionF, 76
positionF2, 76
positionF2_x, 76
positionF3, 76
positionF_x, 76
positionL_x, 77
positionR1, 77
positionR2, 77
setController, 71
speed_cloud, 77
speed_floor, 77
speed_light, 77
speed_little, 77
speedFlakes, 77
speedFlakes2, 78
speedRain, 78
thunder, 78
thunder1, 78
update, 72
value, 78
motion
AllegroController, 12
movement
PipePair, 95
Music, 78
∼Music, 79
Music, 79
music, 80
playMusic, 80
music
Music, 80
musicState
SettingsMenu, 113

name
Button, 23
Player, 107
nameCampDeselect
LoadName, 63
nameCampSelect
LoadName, 63
nameFont
LoadName, 63
noError
load_game.hpp, 144
noexistName
load_game.hpp, 144
noName
load_game.hpp, 144

obj_sprite
GameObject, 49
operator<
Player, 104
operator<<
player.cpp, 158
OSCILATION
defines.hpp, 129

PAUSE
defines.hpp, 131
Pipe, 80
check_score, 82
draw, 83
get_y, 83
is_off_screen, 83
Pipe, 82
scored, 84
set_vx, 83
set_y, 83
update, 84
vx, 84
pipe
FlappyBird, 41
pipe1
PipeList, 92
PIPE_SPEED
defines.hpp, 130
PIPE_VERTICAL_SPEED
defines.hpp, 130
PipeList, 85
add_pipe_pair, 86
check_collision, 86
check_score, 87
delete_pipe_pair, 88
difficulty_pipe, 91
draw, 88
gen, 91
get_pipe_pairs, 88
get_points, 89
pipe1, 92
PipeList, 86
Pipes, 92
points, 92
reset, 89
set_difficulty, 89
set_start, 90
set_vx, 90
start, 92
update, 91
pipelist
FlappyBird, 42
PipePair, 93
bottom, 94
movement, 95
PipePair, 94
signal, 95
top, 95
Pipes
PipeList, 92

PLAY
defines.hpp, 131
Play, 95
buttonExitDeselect, 99
buttonExitSelect, 99
buttonPause, 99
buttonPositionSelected, 100
buttonTryagainDeselect, 100
buttonTryagainSelect, 100
draw, 97
enter, 97
flappy, 100
font, 100
handle_input, 98
logoGameOver, 100
menuButtons, 100
Play, 97
status, 101
update, 98
Player, 101
CheckingName, 102
GetName, 103
GetScore, 103
name, 107
operator<, 104
Player, 102
ReadLeaderboard, 104
SaveLeaderboard, 105
score, 107
SetScore, 106
ShowLeaderboard, 107
SortLeaderboard, 107
player
init.cpp, 155
init.hpp, 137
player.cpp
operator<<, 158
playerNameString
LoadName, 63
playMusic
Music, 80
playSound
Sound, 115
points
PipeList, 92
pointSound
FlappyBird, 42
positionB_x
Motion, 76
positionC_x
Motion, 76
positionCC_x
Motion, 76
positionF
Motion, 76
positionF2
Motion, 76
positionF2_x
FlappyBird, 42
Motion, 76
positionF3
Motion, 76
positionF_x
FlappyBird, 42
Motion, 76
positionL_x
Motion, 77
positionR1
Motion, 77
positionR2
Motion, 77

queue
State, 118

r
TextFont, 120
ranking
init.cpp, 155
init.hpp, 137
ReadLeaderboard
Player, 104
README.md, 148
reset
FlappyBird, 36
PipeList, 89
reset_xy
Bird, 18
ROTATION
defines.hpp, 130
rotation
Bird, 22
run
AllegroController, 11

saveCurrentPlayerScore
FlappyBird, 36
SaveLeaderboard
Player, 105
score
FlappyBird, 42
Player, 107
scored
Pipe, 84
SCREEN_H
defines.hpp, 130
screen_height
AllegroController, 12
SCREEN_W
defines.hpp, 130
screen_width
AllegroController, 12
ScreenState
defines.hpp, 131
selectSound
assets.cpp, 150
assets.hpp, 126

init.hpp, 137
set_break
Bird, 18
set_by
Bird, 19
set_current_player
FlappyBird, 37
set_difficulty
PipeList, 89
set_finals
GameObject, 48
set_playerscore
FlappyBird, 37
set_start
PipeList, 90
set_vx
Pipe, 83
PipeList, 90
set_x
GameObject, 48
set_y
GameObject, 48
Pipe, 83
setColor
TextFont, 119
setController
Motion, 71
setGlobals
State, 117
SetScore
Player, 106
SettingsMenu, 108
buttonBackDeselect, 111
buttonBackSelect, 111
buttonMusicDaySelect, 112
buttonMusicRainSelect, 112
buttonMusicSelect, 112
buttonMusicSnowSelect, 112
buttonNoMusicDaySelect, 112
buttonNoMusicRainSelect, 112
buttonNoMusicSnowSelect, 112
buttonPositionSelected, 112
campSettingsMusic, 113
campSettingsNoMusic, 113
draw, 109
enter, 110
font, 113
handle_input, 110
menuButtons, 113
musicState, 113
SettingsMenu, 109
update, 111
weatherSelected, 113
ShowLeaderboard
Player, 107
signal
PipePair, 95
SortLeaderboard
Player, 107
Sound, 114
∼Sound, 115
playSound, 115
Sound, 114
sound, 115
sound
Sound, 115
speed_cloud
Motion, 77
speed_floor
Motion, 77
speed_light
Motion, 77
speed_little
Motion, 77
speedFlakes
Motion, 77
speedFlakes2
Motion, 78
speedRain
Motion, 78
src/allegro_interface.cpp, 148
src/assets.cpp, 148
src/bird.cpp, 150
src/flappy_bird_controller.cpp, 150
src/game_object.cpp, 151
src/init.cpp, 151
src/main.cpp, 155
src/motion.cpp, 156
src/pipe.cpp, 157
src/player.cpp, 157
src/state.cpp, 158
src/states/difficulty_menu.cpp, 159
src/states/leaderboard_menu.cpp, 159
src/states/load_game.cpp, 160
src/states/main_menu.cpp, 160
src/states/play.cpp, 161
src/states/settings_menu.cpp, 162
start
PipeList, 92
starter
FlappyBird, 37
State, 116
∼State, 116
display, 118
draw, 117
enter, 117
ev, 118
handle_input, 117
queue, 118
setGlobals, 117
update, 117
state
FlappyBird, 42
status
Play, 101
successInsert

load_game.hpp, 144

TETO_BIRD
defines.hpp, 130
TextFont, 118
~TextFont, 119
b, 120
font, 120
g, 120
r, 120
setColor, 119
TextFont, 119
writeText, 120
thunder
Motion, 78
thunder1
Motion, 78
time
FlappyBird, 42
TIME_GIF_BIRD
defines.hpp, 130
timer_FPS
init.cpp, 155
init.hpp, 137
top
PipePair, 95

unbreaker
FlappyBird, 38
unloadGlobalAssets
assets.cpp, 149
assets.hpp, 125
update
Bird, 19
DifficultyMenu, 26
FlappyBird, 38
GameObject, 49
LeaderboardMenu, 56
LoadName, 61
MainMenu, 67
Motion, 72
Pipe, 84
PipeList, 91
Play, 98
SettingsMenu, 111
State, 117
update_score
FlappyBird, 39

value
Motion, 78
valueB
Bird, 22
velocity
FlappyBird, 43
velocity_backup
FlappyBird, 43
vx
Pipe, 84
vy
Bird, 22

weatherSelected
SettingsMenu, 113
width
GameObject, 49
writeText
TextFont, 120

x
GameObject, 49
Image, 53
x_final
GameObject, 50
X_INIT
defines.hpp, 130

y
GameObject, 50
Image, 53
y_final
GameObject, 50