Function Index by Class — *refactor-ready cheat-sheet*

(• What it does • Called by)

Ball (ball.py)

• __init__(x, y, speed) – create sprite, cache image, randomise velocity
  • Called by → GameBoardManager.upload_sprites

• update_Move(vx, vy) – store new velocity vector (legacy name kept)
  • Called by → BallManager._apply_motion

• get_move() – return current (vx, vy); used by AI predictor
  • Called by → MovementManager._predict_ball_intercept

• reset_position(x, y) – centre ball, randomise launch direction
  • Called by → CollisionManager.handle_side_collision

• _randomize_velocity() – helper for the two methods above

BallManager (BallManager.py)

• init_sounds() – mixer init + load bounce / score SFX once
  • Called by → pongGame.main

• move_balls() (**alias: ball_move**) – per-frame: move all balls, collide, redraw
  • Called by → pongGame.game_loop

• _move_single_ball(ball, idx, …) – full physics cycle for one ball
  • Called by → move_balls

• _apply_motion(ball, vx, vy) – write velocity, advance pos, blit sprite
  • Called by → _move_single_ball

• _redraw_screen() – clear BG, centre-line, sprites, score, flip
  • Called by → move_balls

• play_sound(snd) – queue a sound on free mixer channel
  • Called by → CollisionManager, _move_single_ball

CollisionManager (CollisionManager.py)

• handle_paddle_collision(ball, left, right, …) – detect paddle hit, reflect, add speed
  • Called by → BallManager._move_single_ball

• handle_wall_collision(y, vy, h) – bounce off top/bottom walls
  • Called by → BallManager._move_single_ball

• handle_side_collision(ball, idx, x, sfx) – ball leaves field → score + reset
  • Called by → BallManager._move_single_ball

• check_game_over() – show winner screen & wait for restart when lives = 0
  • Called by → handle_side_collision, _move_single_ball

(plus several small "_" helpers: _wait_for_restart, _center_ball, _update_lives_after_score, _overlaps_paddle, _play_sound)

GameBoardManager (GameBoardManager.py)

• upload_screen(title, bg, accent) – create window, draw background, centre line
  • Called by → pongGame.start_game

• upload_sprites() – spawn paddles & balls, first draw
  • Called by → pongGame.start_game

• r_screen(bg, line) – full per-frame redraw used by BallManager
  • Called by → BallManager._redraw_screen

• clear_screen, redraw, _draw_highlights – internal helpers

GameTextManager (GameTextManager.py)

• draw_text(…) – quick font-render + optional display.flip
• font_width(…) – cached font-measure helper
• draw_gradient_background – animated menu background
• draw_settings_background – blue→white BG behind settings panel
• draw_menu – opponent-select menu renderer
• show_score – draw lives HUD
• game_over_message(winner) – black screen + winner text
• draw_center_line – dashed divider
• draw_button – rounded button with centred label
  • Called by various UI / HUD functions across modules

InputBox (InputBox.py)

• __init__ – build numeric entry box
• handle_event(event) – edit text / toggle focus
• draw(screen) – render label, box, text, coloured border
• is_valid() – digit & within range?
• get_value() – int or None
  • Called by → game_settings

Module-level game_settings() – full-screen dialog, returns on "Let's Play!"
  • Called by → pongGame.main

MovementManager (MovementManager.py)

• _move_player(event, paddle, keymap) – human keypress → paddle jump
• _predict_ball_intercept(ball) – foresee y + time ball hits AI paddle
• _get_best_ball_target() – choose first-arriving ball (multi-ball)
• _move_ai(paddle, y_target) – PID-like move with difficulty presets
• sprite_movement(event=None) – call each frame; handles keys + AI

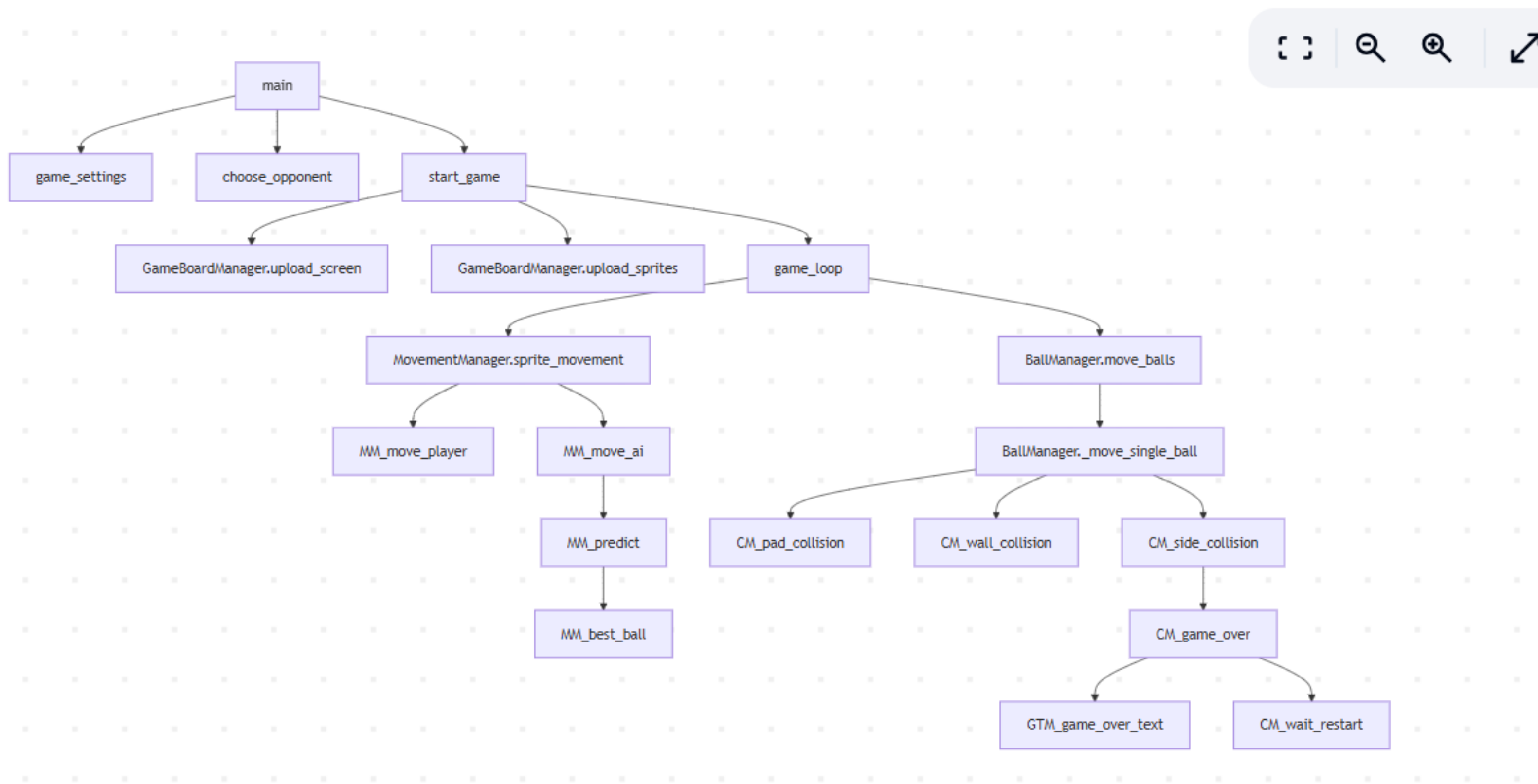
  - Called by → pongGame.game_loop (with and without KEYDOWN)
- game_loop_movement(clock) – legacy continuous AI tick (kept for compat)

Player (player.py)

- __init__(x, y, lives) – load sprite, centre vertically, set lives
- clamp_y(y) / constrain_y() – keep paddle inside screen
- lose_life(side) – decrement side's life counter
- get_life() – return (leftLives, rightLives)
  - Called by → CollisionManager, GameTextManager.show_score

Module-level drivers

InputBox.py – game_settings() – collects balls / lives / AI level
pongGame.py –
- main() → init pygame, settings, opponent, start game
- start_game() → board upload + sprites, then game_loop()
- choose_opponent() → animated C/F menu, returns bool
- game_loop() → infinite frame loop (events → MovementManager / BallManager)


main
game_settings
choose_opponent
start_game
GameBoardManager.upload_screen
GameBoardManager.upload_sprites
game_loop
MovementManager.sprite_movement
BallManager.move_balls
MM_move_player
MM_move_ai
BallManager._move_single_ball
MM_predict
CM_pad_collision
CM_wall_collision
CM_side_collision
MM_best_ball
CM_game_over
GTM_game_over_text
CM_wait_restart