tetris-pygame

Release 0.0.1

Łukasz Łukaszewski

Dec 03, 2023

CONTENTS

CHAPTER

ONE

MAIN MODULE

`main.`**`main`**`()` → None

Main function of the game, creates the settings, menu and game objects and runs main functions of them in the loop

CHAPTER

TWO

SETTINGS MODULE

class settings.**Settings**

Bases: object

Settings class for the game

BG_COLOR: tuple = (0, 0, 100)

CELL_BORDER_COLOR: tuple = (119, 136, 153)

CHECK_KEYS_PRESSED_MOVEMENT_DOWN_TIME: int = 50

CHECK_KEYS_PRESSED_MOVEMENT_SIDE_TIME: int = 80

CHECK_KEYS_PRESSED_ROTATION_TIME: int = 150

CONTROLS_ONE_KEY_X: float = 650.0

CONTROLS_TEXTS: list[str] = ['move down', 'move left', 'move right', 'hard drop', 'rotate left', 'rotate right']

CONTROLS_TEXTS_X: float = 333.3333333333333

CONTROLS_TEXTS_Y: list[float] = [320.0, 400.0, 480.0, 560.0, 640.0, 720.0]

CONTROLS_TITLE: str = 'CONTROLS'

CONTROLS_TITLE_Y: int = 160

CONTROLS_TWO_KEYS_X: list[float] = [600.0, 700.0]

EMPTY_CELL_TAG: int = 0

END_OF_BTNS_COLOR: tuple = (218, 165, 32)

END_OF_GAME_BTNS_HEIGHT: int = 44

END_OF_GAME_BTNS_WIDTH: int = 125

END_OF_GAME_BTNS_Y: int = 400

END_OF_GAME_MENU_BTN_TEXT: str = 'Menu'

END_OF_GAME_MENU_BTN_X: int = 345

END_OF_GAME_NEXT_BTN_TEXT: str = 'Next'

END_OF_GAME_NEXT_BTN_X: int = 530

FONT_COLOR: tuple = (255, 255, 255)

FONT_NAME: str = 'Tahoma'

FONT_SIZE_CONTROLS: int = 30

FONT_SIZE_CONTROLS_TITLE: int = 40

FONT_SIZE_END_OF_GAME_BTNS: int = 20

FONT_SIZE_GET_USERNAME: int = 23

FONT_SIZE_INFO_TITLES: int = 30

FONT_SIZE_LEADERBOARD: int = 15

FONT_SIZE_LEADERBOARD_HEADER: int = 20

FONT_SIZE_LEADERBOARD_TITLE: int = 40

FONT_SIZE_RULES: int = 30

FONT_SIZE_RULES_TITLE: int = 40

FONT_SIZE_SCORE_LVL: int = 30

FONT_SIZE_TETRIS_TITLE: int = 80

FPS: int = 60

GAME_BORDER_COLOR: tuple = (255, 0, 0)

GAME_WINDOW_HEIGHT: int = 600

GAME_WINDOW_WIDTH: int = 400

GET_USERNAME_INPUT_BOX_HEIGHT: int = 40

GET_USERNAME_INPUT_BOX_WIDTH: int = 250

GET_USERNAME_INPUT_BOX_X: float = 588.2352941176471

GET_USERNAME_INPUT_BOX_Y: float = 392.0

GET_USERNAME_TEXT: str = 'ENTER YOUR USERNAME (ENTER)'

GO_BACK_BTN_X: int = 30

GO_BACK_BTN_Y: int = 40

GO_BACK_ICON_FILENAME: str = 'assets/arrow_back.png'

GRID_CELL_HEIGHT: int = 30

GRID_CELL_WIDTH: int = 40

GRID_N_OF_COL: int = 10

GRID_N_OF_ROWS: int = 22

HARD_DROP_KEY_FILENAME: str = 'assets/space_key.png'

HARD_DROP_LOOP_SLEEP_TIME: float = 0.01

I: tuple = (0, 255, 255)

INFO_WINDOW_HEIGHT: int = 150

INFO_WINDOW_WIDTH: int = 219

J: tuple = (0, 0, 255)

L: tuple = (255, 165, 0)

LEADERBOARD_BORDER_COLOR: tuple = (128, 128, 128)

LEADERBOARD_BORDER_WIDTH: int = 1

LEADERBOARD_BORDER_X: float = 62.5

LEADERBOARD_FIRST_ROW_Y: float = 352.22222222222223

LEADERBOARD_HEADERS_TEXTS: list[str] = ['RANK', 'USERNAME', 'SCORE', 'LVL', 'GAMES PLAYED']

LEADERBOARD_HEADER_BORDER_WIDTH: int = 3

LEADERBOARD_HEADER_HEIGHT: int = 100

LEADERBOARD_HEADER_TEXT_Y: float = 272.22222222222223

LEADERBOARD_HEADER_Y: float = 222.22222222222223

LEADERBOARD_ROW_HEIGHT: float = 40.0

LEADERBOARD_TEXT_WIDTH: float = 175.0

LEADERBOARD_TEXT_X: list[float] = [107.14285714285715, 282.14285714285717, 500.0, 675.0, 850.0]

LEADERBOARD_TITLE: str = 'LEADERBOARD'

LEADERBOARD_TITLE_Y: int = 160

LEADERBOARD_WIDTH: float = 875.0

LVL_TITLE: str = 'LEVEL'

LVL_TITLE_Y: float = 410.2564102564103

LVL_WINDOW_X: int = 40

LVL_WINDOW_Y: float = 444.44444444444446

MENU_BTNS_FIRST_Y: int = 200

MENU_BTNS_HEIGHT: int = 100

MENU_BTN_GAP: int = 20

MENU_CONTROLS_TEXT = 'Controls'

MENU_LEADERBOARD_TEXT = 'Leaderboard'

MENU_QUIT_TEXT = 'Quit'

MENU_RULES_TEXT = 'Rules'

MENU_START_GAME_TEXT = 'Start game'

MOVE_DOWN_ACCELERATION_PER_LVL: int = 19

MOVE_DOWN_START_TIME: int = 1000

MOVING_DOWN_KEY_1_FILENAME: str = 'assets/s_key.png'

MOVING_DOWN_KEY_2_FILENAME: str = 'assets/down_key.png'

MOVING_LEFT_KEY_1_FILENAME: str = 'assets/a_key.png'

MOVING_LEFT_KEY_2_FILENAME: str = 'assets/left_key.png'

MOVING_RIGHT_KEY_1_FILENAME: str = 'assets/d_key.png'

MOVING_RIGHT_KEY_2_FILENAME: str = 'assets/right_key.png'

NEXT_TETROMINO_CELL_HEIGHT: int = 37

NEXT_TETROMINO_CELL_WIDTH: int = 54

NEXT_TETROMINO_N_OF_COL: int = 4

NEXT_TETROMINO_N_OF_ROWS: int = 4

NEXT_TETROMINO_TITLE: str = 'NEXT'

NEXT_WINDOW_X: float = 740.7407407407406

NEXT_WINDOW_Y: int = 200

N_OF_LINES_TO_LVL_UP: int = 10

O: tuple = (255, 255, 0)

POINTS_FOR_HARD_DROP: int = 2

POINTS_FOR_SOFT_DROP: int = 1

POINTS_PER_LINES: dict = {1: 100, 2: 300, 3: 500, 4: 800}

ROTATE_LEFT_KEY_FILENAME: str = 'assets/z_key.png'

ROTATE_RIGHT_KEY_1_FILENAME: str = 'assets/w_key.png'

ROTATE_RIGHT_KEY_2_FILENAME: str = 'assets/up_key.png'

RULES_TEXT_Y: float = 320.0

RULES_TITLE: str = 'RULES'

RULES_TITLE_Y: int = 160

`S: tuple = (0, 255, 0)`

`SCORE_NEXT_TITLE_Y: int = 160`

`SCORE_TITLE: str = 'SCORE'`

`SCORE_WINDOW_X: int = 40`

`SCORE_WINDOW_Y: int = 200`

`SCREEN_HEIGHT: int = 800`

`SCREEN_WIDTH: int = 1000`

`SECOND_BG_COLOR: tuple = (0, 0, 0)`

`T: tuple = (128, 0, 128)`

`TETRIS_TITLE: str = 'TETRIS'`

`TETRIS_TITLE_Y: int = 50`

`Z: tuple = (255, 0, 0)`

check_go_back_btn_hover() → None

Checks if mouse is hovering over the go_back button and changes the cursor accordingly

create_end_of_game_btns() → None

Creates buttons displayed after the game is lost

create_get_username_text() → None

Creates the input box for the username

create_go_back_btn() → None

Creates the go back button

create_titles_properties() → None

Create properties for the titles

draw_go_back_btn() → None

Draws the go back button

draw_tetris_title() → None

Draws the tetris title

CHAPTER

THREE

MENU MODULE

class menu.**Menu**(*settings:* Settings)

Bases: `object`

Menu class for the game

CONTROLS_CHOICE: str = 'controls'

GAME_CHOICE: str = 'game'

LEADERBOARD_CHOICE: str = 'leaderboard'

QUIT_CHOICE: str = 'quit'

RULES_CHOICE: str = 'rules'

check_events() → bool | None

Checks for pygame events QUIT and MOUSEBUTTONDOWN and reacts to them

Returns
True if game, leaderboard, controls or rules button is pressed, None otherwise

Return type
(bool | None)

check_hover() → None

Checks if mouse is hovering over the buttons and changes the cursor accordingly

choice: str = ''

create_buttons() → None

Creates menu buttons and its rects

draw_buttons() → None

Draws the buttons on the screen

main() → str

Main menu loop, waits for the user to press a button and returns the choice

Returns
Choice of the user

Return type
(str)

CHAPTER

FOUR

GAME MODULE

class game.**Game**(*settings: Settings*)

Bases: object

Class contains main game logic and methods to draw game elements

add_score(*lines_cleared: int*) → None

Add score for cleared lines

check_events() → None

Check pygane events and react to them

check_hover() → None

Checks if mouse is hovering over the buttons and changes the cursor accordingly

check_line() → bool

Check if there is a line of tetrominos and delete it

Parameters

None –

Returns

True if there is a line of tetrominos, False otherwise

Return type

(bool)

check_lvl_up() → None

check_pressed_down_movement() → None

Check if down movement keys are pressed and react to them

check_pressed_rotate() → None

Check if rotation keys are pressed and react to them

check_pressed_side_movement() → None

Check if side movement keys are pressed and react to them

check_tetromino_above_top() → bool

Check if there is a tetromino above visible top of the grid

create_game_windows() → None

Create game windows

db_insert_user() → None

Insert user into database

`delete_line`(*row*) → None

Delete line from grid and move all tetrominos above it down

`draw_end_of_game_btns`() → None

Draw buttons displayed at the end of the game, after loss

`draw_game_window`() → None

Draw game window on screen

`draw_grid`() → None

Draw grid with tetrominos on game window

`draw_lvl`() → None

Draw score window and score on

`draw_lvl_title`() → None

Draw score title on screen

`draw_lvl_window`() → None

Draw score window on screen

`draw_next_tetromino`() → None

Draw next tetromino on next tetromino window

`draw_next_tetromino_title`() → None

Draw next tetromino title on screen

`draw_next_tetromino_window`() → None

Draw next tetromino window on screen

`draw_score`() → None

Draw score window and score on

`draw_score_title`() → None

Draw score title on screen

`draw_score_window`() → None

Draw score window on screen

`game_window: Surface`

`game_window_rect: Rect`

`get_username`() → None

`grid: list[list[int]]`

`init_properties`() → None

Initialize game properties

`lines_cleared: int`

`lvl: int`

`lvl_up`() → None

`lvl_window: Surface`

`lvl_window_rect: Rect`

main() → None

Main game loop

move_down_key_pressed: bool = False

next_game() → bool

Check if user clicked on next game button or menu button

next_tetromino_grid: list[list[int]]

next_tetromino_window: Surface

next_tetromino_window_rect: Rect

print_grid() → None

Print grid in console

random_tetromino() → *Tetromino*

Return random tetromino

Returns

Random tetromino

Return type

(*Tetromino*)

score: int

score_window: Surface

score_window_rect: Rect

space_down: bool = False

username: str

CHAPTER

FIVE

TETROMINOS MODULE

class tetrominos.**Itetromino**(*game: Game*)

Bases: *Tetromino*

NEXT_TETROMINO_GRID_POS: list[list] = [[1, 0], [1, 1], [1, 2], [1, 3]]

SPAWN_POS: list[list] = [[1, 3], [1, 4], [1, 5], [1, 6]]

TAG: str = 'I'

pos0() → None

Rotates the tetromino to position 0

pos1() → None

Rotates the tetromino to position 1

pos2() → None

Rotates the tetromino to position 2

pos3() → None

Rotates the tetromino to position 3

rotate_left() → None

Rotates the tetromino left

rotate_right() → None

Rotates the tetromino right

class tetrominos.**Jtetromino**(*game: Game*)

Bases: *Tetromino*

NEXT_TETROMINO_GRID_POS: list[list] = [[1, 0], [2, 0], [2, 1], [2, 2]]

SPAWN_POS: list[list] = [[0, 3], [1, 4], [1, 3], [1, 5]]

TAG: str = 'J'

pos0() → None

Set the tetromino to position 0

pos1() → None

Set the tetromino to position 1

pos2() → None

Set the tetromino to position 2

`pos3()` → None

Set the tetromino to position 3

`class tetrominos.Ltetromino(`*game:* Game`)`

Bases: *`Tetromino`*

`NEXT_TETROMINO_GRID_POS: list[list] = [[2, 0], [2, 1], [2, 2], [1, 2]]`

`SPAWN_POS: list[list] = [[1, 3], [1, 4], [1, 5], [0, 5]]`

`TAG: str = 'L'`

`pos0()` → None

Set the tetromino to position 0

`pos1()` → None

Set the tetromino to position 1

`pos2()` → None

Set the tetromino to position 2

`pos3()` → None

Set the tetromino to position 3

`class tetrominos.Otetromino(`*game:* Game`)`

Bases: *`Tetromino`*

`NEXT_TETROMINO_GRID_POS: list[list] = [[1, 1], [2, 1], [1, 2], [2, 2]]`

`SPAWN_POS: list[list] = [[1, 4], [1, 5], [0, 4], [0, 5]]`

`TAG: str = 'O'`

`rotate_left()` → None

Do nothing because the tetromino is a square

`rotate_right()` → None

Do nothing because the tetromino is a square

`class tetrominos.Stetromino(`*game:* Game`)`

Bases: *`Tetromino`*

`NEXT_TETROMINO_GRID_POS: list[list] = [[2, 0], [2, 1], [1, 1], [1, 2]]`

`SPAWN_POS: list[list] = [[1, 3], [1, 4], [0, 4], [0, 5]]`

`TAG: str = 'S'`

`pos0()` → None

Set the tetromino to position 0

`pos1()` → None

Set the tetromino to position 1

`pos2()` → None

Set the tetromino to position 2

`pos3()` → None

Set the tetromino to position 3

class `tetrominos.`**`Tetromino`**(*game: Game*)

Bases: `object`

Tetromino class for the game

`LAST_COL_IDX: int = 9`

`NEXT_TETROMINO_GRID_POS: list[list]`

`SPAWN_POS: list[list]`

`TAG: str`

`check_cell_available_for_rotation`(*cell: list[int]*) → bool

Check if the cell is available for rotation

Parameters

cell (*list[int]*) – Cell to check

Returns

True if the cell is available for rotation, False otherwise

Return type

(bool)

`check_down`() → bool

Checks if the tetromino is at the bottom of the grid

Returns

True if the tetromino is at the bottom of the grid, False otherwise

Return type

(bool)

`check_move_left`() → bool

Check if the tetromino can move left by checking if it is at the left edge of the grid or if it is touching another tetromino on the left

Returns

True if the tetromino can move left, False otherwise

Return type

(bool)

`check_move_right`() → bool

Check if the tetromino can move right by checking if it is at the right edge of the grid or if it is touching another tetromino on the right

Returns

True if the tetromino can move right, False otherwise

Return type

(bool)

`check_touch`() → bool

Check if the bottom of the tetromino is touching another tetromino

Returns

True if the bottom of the tetromino is touching another tetromino, False otherwise

Return type

(bool)

clear() → None

Clears the grid from the tetromino

clear_next_tetromino_window() → None

Clears the next tetromino window

hard_drop() → None

Hard drops the tetromino

move_down() → bool | None

Moves the tetromino down

move_left() → None

Moves the tetromino left

move_right() → None

Moves the tetromino right

pos: list[list]

pos0() → None

Rotates the tetromino to position 0

pos1() → None

Rotates the tetromino to position 1

pos2() → None

Rotates the tetromino to position 2

pos3() → None

Rotates the tetromino to position 3

put_on_next_tetromino_window() → None

Puts the next tetromino on the next tetromino window

rotate_left() → None

Rotates the tetromino left

rotate_right() → None

Rotates the tetromino right

spawn() → None

Set the spawn position

update_on_grid() → None

Updates the grid with the new position of the tetromino

class tetrominos.**Ttetromino**(*game:* Game)

Bases: *Tetromino*

NEXT_TETROMINO_GRID_POS: list[list] = [[2, 0], [2, 1], [2, 2], [1, 1]]

SPAWN_POS: list[list] = [[1, 2], [1, 3], [1, 4], [0, 3]]

TAG: str = 'T'

pos0() → None

Set the tetromino to position 0

pos1() → None

Set the tetromino to position 1

pos2() → None

Set the tetromino to position 2

pos3() → None

Set the tetromino to position 3

class tetrominos.**Ztetromino**(*game:* Game)

Bases: *Tetromino*

NEXT_TETROMINO_GRID_POS: list[list] = [[1, 0], [1, 1], [2, 1], [2, 2]]

SPAWN_POS: list[list] = [[0, 3], [1, 4], [0, 4], [1, 5]]

TAG: str = 'Z'

pos0() → None

Set the tetromino to position 0

pos1() → None

Set the tetromino to position 1

pos2() → None

Set the tetromino to position 2

pos3() → None

Set the tetromino to position 3

CHAPTER

SIX

LEADERBOARD MODULE

class leaderboard.**Leaderboard**(*settings: Settings*)

Bases: `object`

Class for the leaderboard screen

check_events() → bool | None

Checks for events

Returns
True if go back button is pressed, None otherwise

Return type
(bool | None)

create_draw_row(*rank: int*, *user: User*) → None

Creates and draws a row of the leaderboard

create_header() → None

Creates the header of the leaderboard

create_leaderboard() → None

Creates the leaderboard

create_title() → None

Creates the title of the leaderboard

draw_header() → None

Draws the leaderboard header

draw_title() → None

Draws the leaderboard title

main() → None

Main function of the leaderboard screen that draws everything and checks for events

CHAPTER

SEVEN

CONTROLS MODULE

class controls.**Controls**(*settings: Settings*)

Bases: `object`

Class for the controls screen

check_events() → bool | None

Checks for events

Returns
True if go back button is pressed, None otherwise

Return type
(bool | None)

create_controls() → None

Creates the controls

create_title() → None

Creates the title of the controls

draw_controls() → None

Draws the controls

draw_title() → None

Draws the controls title

main() → None

Main method for the controls screen, draws everything and checks for events

CHAPTER

EIGHT

RULES MODULE

`class rules.`**`Rules`**(*settings:* Settings)

Bases: `object`

A class to show the rules of the game

`TEXT: str = 'The goal of the game is to get the highest score\npossible. You earn points by moving down tetrominos,\n1 point for soft drop and two points for hard drop.\nYou can also earn points by clearing lines.\n1 line is 100 points, 2 lines is 300 points,\n3 lines is 500 points, and 4 lines is 800 points.\nEach 10 lines cleared, the level increases.\nThe level increases the speed of the tetrominos.\nThe game ends when the tetrominos reach\nthe top of the screen.'`

`check_events``()` → bool | None

Checks for events

Returns
True if go back button is pressed, None otherwise

Return type
(bool | None)

`create_text``()` → None

Creates the text of the rules

`create_title``()` → None

Creates the title of the controls

`draw_text``()` → None

`draw_title``()` → None

Draws the controls title

`main``()` → None

Main method for the controls screen

CHAPTER

NINE

DB

9.1 models user module

class db.models.user.**User**(**args*, ***kwargs*)

Bases: `Model`

Model for user table

DoesNotExist

alias of `UserDoesNotExist`

games_played = <IntegerField: User.games_played>

highest_score = <IntegerField: User.highest_score>

id = <AutoField: User.id>

lvl = <IntegerField: User.lvl>

username = <CharField: User.username>

db.models.user.**on_save_handler**(*model_class*, *instance*, *created*) → None

Increments games_played field on save

db.models.user.**user_exists**(*username*) → bool

Checks if a user exists in the database

9.2 scripts migration model

db.scripts.migration.**create_user_table**() → None

Create the user table

CHAPTER

TEN

TEST

10.1 test game module

class tests.test_game.**TestGame**(*methodName='runTest'*)

Bases: TestCase

CHECK_LINE_FALSE_GRID = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ['I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I'], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

CHECK_LINE_TRUE_GRID = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ['I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I'], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

CHECK_TETROMINO_GRID = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ['I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I'], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

CLEAR_GRID = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

CLEAR_NEXT_TETROMINO_GRID = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

DELETE_LINE_AFTER_GRID = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 'L', 0, 0, 0, 0], [0, 0, 0, 'L', 'L', 'L', 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 'O', 'O', 0, 0, 0, 0], ['I', 'I', 'I', 'I', 'O', 'O', 0, 0, 0, 0], [0, 'T', 0, 0, 0, 'Z', 0, 'J', 'J', 0], [0, 'T', 'T', 0, 'Z', 'Z', 0, 'J', 'O', 'O'], [0, 'T', 0, 0, 'Z', 0, 0, 'J', 'O', 'O'], ['Z', 'Z', 0, 'O', 'O', 0, 0, 0, 0, 'L'], [0, 'Z', 'Z', 'O', 'O', 'L', 0, 'L', 'L', 'L'], [0, 'Z', 'Z', 'L', 'L', 'L', 0, 'Z', 'Z', 0], ['O', 'O', 0, 'O', 'O', 0, 0, 0, 0, 0], [0, 0, 0, 'O', 'O', 0, 0, 0, 0, 0], [0, 0, 0, 0, 'S', 'S', 0, 0, 0, 0], [0, 0, 0, 'S', 'S', 0, 0, 'O', 'O', 0], [0, 0, 'L', 'O', 'O', 0, 0, 'O', 'O', 'L'], ['L', 'L', 'L', 'O', 'O', 0, 0, 'L', 'L', 'L']]

DELETE_LINE_BEFORE_GRID = [[0, 0, 0, 0, 0, 'L', 0, 0, 0, 0], [0, 0, 0, 'L', 'L', 'L', 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 'O', 'O', 0, 0, 0, 0], ['I', 'I', 'I', 'I', 'O', 'O', 0, 0, 0, 0], [0, 'T', 0, 0, 0, 'Z', 0, 'J', 'J', 0], [0, 'T', 'T', 0, 'Z', 'Z', 0, 'J', 'O', 'O'], [0, 'T', 0, 0, 'Z', 0, 0, 'J', 'O', 'O'], ['Z', 'Z', 0, 'O', 'O', 0, 0, 0, 0, 'L'], [0, 'Z', 'Z', 'O', 'O', 'L', 0, 'L', 'L', 'L'], [0, 'Z', 'Z', 'L', 'L', 'L', 0, 'Z', 'Z', 0], ['O', 'O', 'Z', 'Z', 'I', 'I', 'I', 'I', 'Z', 'Z'], ['O', 'O', 0, 'O', 'O', 0, 0, 0, 0, 0], [0, 0, 0, 'O', 'O', 0, 0, 0, 0, 0], [0, 0, 0, 0, 'S', 'S', 0, 0, 0, 0], [0, 0, 0, 'S', 'S', 0, 0, 'O', 'O', 0], [0, 0, 'L', 'O', 'O', 0, 0, 'O', 'O', 'L'], ['L', 'L', 'L', 'O', 'O', 0, 0, 'L', 'L', 'L']]

classmethod setUpClass() → None

Hook method for setting up class fixture before running tests in the class.

test_check_line() → None

test_check_tetromino_above_top() → None

test_delete_line() → None

test_init_properties() → None

test_random_tetromino() → None

10.2 test menu module

class `tests.test_menu.`**`TestMenu`**(*methodName='runTest'*)

Bases: `TestCase`

`setUp`() → None

Hook method for setting up the test fixture before exercising it.

`test_main`(*mock_display_update*, *mock_draw_buttons*, *mock_check_hover*, *mock_check_events*) → None

10.3 test tetrominos module

class `tests.test_tetrominos.`**`TestTetromino`**(*methodName='runTest'*)

Bases: `TestCase`

`CHECK_DOWN_POS` `= [[21, 3], [21, 4], [21, 5], [21, 6]]`

`LEFT_EDGE_POS` `= [[1, 0], [1, 1], [1, 2], [1, 3]]`

`RIGHT_EDGE_POS` `= [[1, 9], [1, 8], [1, 7], [1, 6]]`

classmethod **`setUpClass`**() → None

Hook method for setting up class fixture before running tests in the class.

`test_check_down`() → None

`test_check_move_left`() → None

`test_check_move_right`() → None

`test_check_touch`() → None

CHAPTER ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX