

```

import pygame
import random

"""
10 x 20 square grid
shapes: S, Z, I, O, J, L, T
represented in order by 0 - 6
"""

pygame.font.init()

# GLOBALS VARS
s_width = 800
s_height = 700
play_width = 300 # meaning 300 // 10 = 30 width per block
play_height = 600 # meaning 600 // 20 = 20 height per block
block_size = 30
5
top_left_x = (s_width - play_width) // 2
top_left_y = s_height - play_height

# SHAPE FORMATS

S = [[['.....',
        '.....',
        '..00.',
        '.00..',
        '.....'],
      [['.....',
        '.....',
        '..0..',
        '..00.',
        '.....']]]

Z = [[['.....',
        '.....',
        '.00..',
        '..00.',
        '.....'],
      [['.....',
        '.....',
        '..0..',
        '..00.',
        '.0...']]]

I = [[['..0..',
        '..0..',
        '..0..',
        '..0..',
        '.....'],
      [['.....',
        '0000.',
        '.....',
        '.....',
        '.....']]]

O = [[['.....',
        '.....',
        '.....',
        '.....',
        '.....']]]

```

```
'..00..' ,
'.00..' ,
'.....' ]]
```

```
J = [[ '.....' ,
        '.0..' ,
        '.000.' ,
        '.....' ],
       [ '.....' ,
        '..00.' ,
        '.0..' ,
        '.0..' ,
        '.....' ],
       [ '.....' ,
        '.000.' ,
        '.....' ],
       [ '.....' ,
        '.0..' ,
        '.0..' ,
        '.00.' ,
        '.....' ]]
```

```
L = [[ '.....' ,
        '.0..' ,
        '.000.' ,
        '.....' ],
       [ '.....' ,
        '..00.' ,
        '.0..' ,
        '.0..' ,
        '.00.' ,
        '.....' ],
       [ '.....' ,
        '.000.' ,
        '.0..' ,
        '.....' ],
       [ '.....' ,
        '.00.' ,
        '.0..' ,
        '.0..' ,
        '.....' ]]
```

```
T = [[ '.....' ,
        '.0..' ,
        '.000.' ,
        '.....' ],
       [ '.....' ,
        '.0..' ,
        '..00.' ,
        '.0..' ,
        '.....' ],
       [ '.....' ,
        '.000.' ,
        '.....' ]]
```

```

'...0..',
'.....'],
['.....',
'...0..',
'.00..',
'...0..',
'.....']]

```

```

shapes = [S, Z, I, O, J, L, T]
shape_colors = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0), (255, 165, 0), (0, 0, 255), (128, 0, 128)]
# index 0 - 6 represent shape

```

```

class Piece(object):
    rows = 20 # y
    columns = 10 # x

    def __init__(self, column, row, shape):
        self.x = column
        self.y = row
        self.shape = shape
        self.color = shape_colors[shapes.index(shape)]
        self.rotation = 0 # number from 0-3

def create_grid(locked_positions={}):
    grid = [[(0,0,0) for x in range(10)] for x in range(20)]

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            if (j,i) in locked_positions:
                c = locked_positions[(j,i)]
                grid[i][j] = c
    return grid

def convert_shape_format(shape):
    positions = []
    format = shape.shape[shape.rotation % len(shape.shape)]

    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            if column == '0':
                positions.append((shape.x + j, shape.y + i))

    for i, pos in enumerate(positions):
        positions[i] = (pos[0] - 2, pos[1] - 4)

    return positions

def valid_space(shape, grid):
    accepted_positions = [[(j, i) for j in range(10) if grid[i][j] == (0,0,0)] for i in range(20)]
    accepted_positions = [j for sub in accepted_positions for j in sub]
    formatted = convert_shape_format(shape)

```

```

for pos in formatted:
    if pos not in accepted_positions:
        if pos[1] > -1:
            return False

return True

def check_lost(positions):
    for pos in positions:
        x, y = pos
        if y < 1:
            return True
    return False

def get_shape():
    global shapes, shape_colors

    return Piece(5, 0, random.choice(shapes))

def draw_text_middle(text, size, color, surface):
    font = pygame.font.SysFont('comicsans', size, bold=True)
    label = font.render(text, 1, color)

    surface.blit(label, (top_left_x + play_width/2 - (label.get_width() / 2),
top_left_y + play_height/2 - label.get_height()/2))

def draw_grid(surface, row, col):
    sx = top_left_x
    sy = top_left_y
    for i in range(row):
        pygame.draw.line(surface, (128,128,128), (sx, sy+ i*30), (sx + play_width,
sy + i * 30)) # horizontal lines
        for j in range(col):
            pygame.draw.line(surface, (128,128,128), (sx + j * 30, sy), (sx + j *
30, sy + play_height)) # vertical lines

def clear_rows(grid, locked):
    # need to see if row is clear the shift every other row above down one

    inc = 0
    for i in range(len(grid)-1, -1, -1):
        row = grid[i]
        if (0, 0, 0) not in row:
            inc += 1
            # add positions to remove from locked
            ind = i
            for j in range(len(row)):
                try:
                    del locked[(j, i)]
                except:
                    continue
    if inc > 0:
        for key in sorted(list(locked), key=lambda x: x[1])[::-1]:
            x, y = key

```

```

        if y < ind:
            newKey = (x, y + inc)
            locked[newKey] = locked.pop(key)

def draw_next_shape(shape, surface):
    font = pygame.font.SysFont('comicsans', 30)
    label = font.render('Next Shape', 1, (255,255,255))

    sx = top_left_x + play_width + 50
    sy = top_left_y + play_height/2 - 100
    format = shape.shape[shape.rotation % len(shape.shape)]

    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            if column == '0':
                pygame.draw.rect(surface, shape.color, (sx + j*30, sy + i*30, 30,
30), 0)

    surface.blit(label, (sx + 10, sy - 30))

def draw_window(surface):
    surface.fill((0,0,0))
    # Tetris Title
    font = pygame.font.SysFont('comicsans', 60)
    label = font.render('TETRIS', 1, (255,255,255))

    surface.blit(label, (top_left_x + play_width / 2 - (label.get_width() / 2),
30))

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            pygame.draw.rect(surface, grid[i][j], (top_left_x + j* 30, top_left_y +
i * 30, 30, 30), 0)

    # draw grid and border
    draw_grid(surface, 20, 10)
    pygame.draw.rect(surface, (255, 0, 0), (top_left_x, top_left_y, play_width,
play_height), 5)
    # pygame.display.update()

def main():
    global grid

    locked_positions = {} # (x,y):(255,0,0)
    grid = create_grid(locked_positions)

    change_piece = False
    run = True
    current_piece = get_shape()
    next_piece = get_shape()
    clock = pygame.time.Clock()
    fall_time = 0
    level_time = 0
    fall_speed = 0.27
    score = 0

```

```

while run:

    grid = create_grid(locked_positions)
    fall_time += clock.get_rawtime()
    level_time += clock.get_rawtime()
    clock.tick()

    if level_time/1000 > 4:
        level_time = 0
        if fall_speed > 0.15:
            fall_speed -= 0.005

    # PIECE FALLING CODE
    if fall_time/1000 >= fall_speed:
        fall_time = 0
        current_piece.y += 1
        if not (valid_space(current_piece, grid)) and current_piece.y > 0:
            current_piece.y -= 1
            change_piece = True

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            pygame.display.quit()
            quit()

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                current_piece.x -= 1
                if not valid_space(current_piece, grid):
                    current_piece.x += 1

            elif event.key == pygame.K_RIGHT:
                current_piece.x += 1
                if not valid_space(current_piece, grid):
                    current_piece.x -= 1
            elif event.key == pygame.K_UP:
                # rotate shape
                current_piece.rotation = current_piece.rotation + 1 %
len(current_piece.shape)
                if not valid_space(current_piece, grid):
                    current_piece.rotation = current_piece.rotation - 1 %
len(current_piece.shape)

            if event.key == pygame.K_DOWN:
                # move shape down
                current_piece.y += 1
                if not valid_space(current_piece, grid):
                    current_piece.y -= 1

            '''if event.key == pygame.K_SPACE:
                while valid_space(current_piece, grid):
                    current_piece.y += 1
                    current_piece.y -= 1
                    print(convert_shape_format(current_piece))''' # todo fix

    shape_pos = convert_shape_format(current_piece)

```

```

# add piece to the grid for drawing
for i in range(len(shape_pos)):
    x, y = shape_pos[i]
    if y > -1:
        grid[y][x] = current_piece.color

# IF PIECE HIT GROUND
if change_piece:
    for pos in shape_pos:
        p = (pos[0], pos[1])
        locked_positions[p] = current_piece.color
    current_piece = next_piece
    next_piece = get_shape()
    change_piece = False

    # call four times to check for multiple clear rows
    if clear_rows(grid, locked_positions):
        score += 10

draw_window(win)
draw_next_shape(next_piece, win)
pygame.display.update()

# Check if user lost
if check_lost(locked_positions):
    run = False

draw_text_middle("You Lost", 40, (255,255,255), win)
pygame.display.update()
pygame.time.delay(2000)

def main_menu():
    run = True
    while run:
        win.fill((0,0,0))
        draw_text_middle('Press any key to begin.', 60, (255, 255, 255), win)
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False

            if event.type == pygame.KEYDOWN:
                main()
    pygame.quit()

win = pygame.display.set_mode((s_width, s_height))
pygame.display.set_caption('Tetris')

main_menu() # start game

```