*import* pygame

*import* random

*from* pygame *import* mixer

*"""*

*10 x 20 square grid*

*shapes: S, Z, I, O, J, L, T*

*represented in order by 0 - 6*

*"""*

*#initialize pygame*

pygame.font.init()

*"""*

*Since in Tetris we have a 10x20 grid,we need to make sure that our play width is exactly half of our play height,so that we have perfect squares.*

*"""*

*# GLOBALS VARS*

s\_width = 800 *# screen width*

s\_height = 700 *# screen height*

play\_width = 300  *# meaning 300 // 10 = 30 width per block*

play\_height = 600  *# meaning 600 // 20 = 20 height per block*

block\_size = 30 *# hence block size = 30*

top\_left\_x = (s\_width - play\_width) // 2

top\_left\_y = s\_height - play\_height

*# SHAPE FORMATS*

S = [['.....',

      '.....',

      '..00.',

      '.00..',

      '.....'],

     ['.....',

      '..0..',

      '..00.',

      '...0.',

      '.....']]

Z = [['.....',

      '.....',

      '.00..',

      '..00.',

      '.....'],

     ['.....',

      '..0..',

      '.00..',

      '.0...',

      '.....']]

I = [['..0..',

      '..0..',

      '..0..',

      '..0..',

      '.....'],

     ['.....',

      '0000.',

      '.....',

      '.....',

      '.....']]

O = [['.....',

      '.....',

      '.00..',

      '.00..',

      '.....']]

J = [['.....',

      '.0...',

      '.000.',

      '.....',

      '.....'],

     ['.....',

      '..00.',

      '..0..',

      '..0..',

      '.....'],

     ['.....',

      '.....',

      '.000.',

      '...0.',

      '.....'],

     ['.....',

      '..0..',

      '..0..',

      '.00..',

      '.....']]

L = [['.....',

      '...0.',

      '.000.',

      '.....',

      '.....'],

     ['.....',

      '..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*

*"""*

*holds information for 'x','y','width','height' for the piece.*

*"""*

*#main data*

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*

        self.lines = 0

*"""*

*creates a 2D list.second list contains colours.creates one list*

*for every row in our grid.The grid is drawn by looping through*

*the 'grid =' which is returned and draws each colour.*

*"""*

*#represents grid #locked\_pos is a dictionary*

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('Rockwell', 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)]

                    pygame.mixer.init()

                    s2 = pygame.mixer.Sound('media.io\_Original Tetris theme (Tetris Soundtrack)-[AudioTrimmer.com].wav')

                    s2.play()

*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)

*return* inc

def draw\_next\_shape(shape, surface, score, last\_score, lines\_cut):

    font = pygame.font.SysFont('Goudy Stout', 20)

    label = font.render('Next Shape', 1, (0,153,0))

    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 - 34, sy- 30))

*#current score*

    font = pygame.font.SysFont('Goudy Stout', 18)

    label = font.render('Score: '+ str(score), 1, (0,153,0))

    sx = top\_left\_x + play\_width + 50

    sy = top\_left\_y + play\_height/2 - 100

    surface.blit(label, (sx - 560, sy - 150))

*#high score*

    font = pygame.font.SysFont('Goudy Stout', 15)

    label = font.render('High Score: '+ last\_score, 1, (0,153,0))

    sx = top\_left\_x + play\_width + 50

    sy = top\_left\_y + play\_height/2 - 100

    surface.blit(label, (sx - 600, sy - 120))

*#lines cut*

    font = pygame.font.SysFont('Goudy Stout', 16)

    label = font.render('Lines Cut: '+ str(lines\_cut), 1, (0,153,0))

    sx = top\_left\_x + play\_width + 50

    sy = top\_left\_y + play\_height/2 - 100

    surface.blit(label, (sx - 580, sy - 90))

def draw\_window(surface, score = 0, last\_score = 0, lines\_cut = 0):

    surface.fill((0,205,205))

*# Tetris Title*

    font = pygame.font.SysFont('Copperplate Gothic', 60, bold = True)

    label = font.render('TETRIS 57', 1, (0,153,0))

    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, (0, 153, 0), (top\_left\_x, top\_left\_y, play\_width, play\_height), 5)

*# pygame.display.update()*

def high\_score(new\_score):

    score = hscore()

*with* open("scores.txt","w") *as* file:

*if* new\_score > int(score):

            file.write(str(new\_score))

*else*:

            file.write(str(score))

def hscore():

*with* open("scores.txt","r") *as* file:

            lines = file.readlines()

            score = lines[0].strip()

*return* score

def main():

    global grid

    last\_score = hscore()

    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

    score = 0

    lines\_cut = 0

*while* run:

        fall\_speed = 0.27

        grid = create\_grid(locked\_positions)

        fall\_time += clock.get\_rawtime()

        clock.tick()

*# 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))

        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*

            score += clear\_rows(grid, locked\_positions)\*10

            lines\_cut += clear\_rows(grid,locked\_positions)

        draw\_window(win,score)

        draw\_next\_shape(next\_piece, win, score, last\_score, lines\_cut)

        pygame.display.update()

*# Check if user lost*

*if* check\_lost(locked\_positions):

            pygame.mixer.stop() *#stops the music*

            run = False

            high\_score(score)

*if* score < 30:

        draw\_text\_middle("Most 7 year olds could do better", 47, (255,0,0), win)

        pygame.display.update()

*#delays time(in miliseconds) till main menu appears*

        pygame.time.delay(2000)

*elif* score <= 50:

        draw\_text\_middle("Alright you're getting there", 47, (255,255,255), win)

        pygame.display.update()

*#delays time(in miliseconds) till main menu appears*

        pygame.time.delay(2000)

*elif* score <= 100:

        draw\_text\_middle("NICE!", 47, (0,204,0), win)

        pygame.display.update()

*#delays time(in miliseconds) till main menu appears*

        pygame.time.delay(2000)

*elif* score <= 500:

        draw\_text\_middle("TETRIS VETERAN", 47, (255,51,127), win)

        pygame.display.update()

*#delays time(in miliseconds) till main menu appears*

        pygame.time.delay(2000)

*elif* score <= 1000:

        draw\_text\_middle("Amin Sahab will be pround", 47, (255,255,255), win)

        pygame.display.update()

*#delays time(in miliseconds) till main menu appears*

        pygame.time.delay(2000)

*else*:

        draw\_text\_middle("YOU'RE TOO OP", 47, (153,0,153), win)

        pygame.display.update()

*#delays time(in miliseconds) till main menu appears*

        pygame.time.delay(2000)

def main\_menu():

    run = True

*while* run:

        win.fill((51,51,255))

        draw\_text\_middle('Press any key to play', 60, (255, 128, 0), win)

        pygame.display.update()

*for* event *in* pygame.event.get():

*if* event.type == pygame.QUIT:

                run = False

*if* event.type == pygame.KEYDOWN:

                pygame.mixer.init()

                s1 = pygame.mixer.Sound('media.io\_Alan Walker - Faded (Instrumental Version).wav')

                s1.play(-1)

                main()

    pygame.quit()

win = pygame.display.set\_mode((s\_width, s\_height))

pygame.display.set\_caption('Tetris 57')

main\_menu()  *# start game*