Mini Project Report submitted in partial fulfillment

of the requirement of OSTL

S. E. (Computer Engineering)

TETRIS

Submitted By

Sr. No	Roll No	Name
1.	18102B2003	Kedar Chendavankar
2	18102B2004	Amey Borole

Under the Guidance of

Prof. Prakash Parmar

Department of Computer Engineering

Vidyalankar Institute of Technology

Wadala(E), Mumbai 400 037

University of Mumbai

2018-19

ABSTRACT

Tetris,mini game created by Russian designer Alexey Pajitnov in 1985 that allows players to rotate falling blocks strategically to clear levels.Tetris has been released for virtually every computer and electronic gaming system, and it is often revered as a classic. Though numerous sequels have been spawned,Tetris games almost always have the same play mechanics: differently shaped blocks drop at varying speeds, and, as the blocks descend, the player must rotate and arrange them to create an uninterrupted horizontal row on the screen. When the player forms one or more solid rows, the completed rows disappear. The goal of the game is to prevent the blocks from stacking up to the top of the screen for as long as possible. Subsequent versions of the game included different modes of play and unique twists, but the overall game play usually mirrored the original tetris quite closely.

Tetris game is the one person game which can be played by the indiviual. It uses pygame library which is more efficient and easy to use than other similar packages. Developing this game is very complex,one should know the basic about python.

This game basically helps the small kids to understand the shapes and colour and also entertain them.

LITERATURE SURVEY

We have included two packages:

1)Pygame: Pygame (**the library**) is a Free and Open Source python programming language library for making multimedia applications like games built on top of the excellent SDL library. Like SDL, pygame is highly portable and runs on nearly every platform and operating system. Millions of people have downloaded pygame itself, which is a whole lot of bits flying across the interwebs. Pygame is a Python wrapper module for the SDL multimedia library. It contains python functions and classes that will allow you to use SDL's support for playing cdroms, audio and video output, and keyboard, mouse and joystick input.

PyGameWidgets is a graphical toolkit for PyGame. It has a grid-based layout manager that allows you to develop great visual complexity through nesting of panels. Widgets can expand using span in a similar fashion to tk. It's routine are merged directly into the pygame namespace.

2)Random: In Python, random module implements pseudo-random number generators for various distributions including integer, float (real). The random module provides access to functions that support these types of operations. Therandom module is another library of functions that can extend the basic features of python. Other modules we have seen so far are string,math,time and graphics. With the exception of the graphics module, all of these modules are built into python. This function generates a list just like range, but then returns one item randomly from that list.

Randint() is an inbuilt function of the random module in Python3. The random module gives access to various useful functions and one of them being able to generate random numbers, which is *randint()*.

CONCULUSION

This was the great experience of developing a mini game-**Tetris** as a mini project**.** Many more to learn about it. It is not so easily understandable in first attempt. While developing this game,one must have the bit knowledge about python datatypes,functions etc.

It can also be developed by using **PyQt** module but the lines of code will be increase.**Pygame** is module which has advanced widgets,functions to create appropriate code. 2D animations is better handled by pygame module.Game programming is very rewarding nowadays and it can also be used in advertising and as a teaching tool too. Game development includes mathematics, logic, physics, AI and much more and it can be amazingly fun. In python, game programming is done in pygame and it is one of the best modules for doing so. We also included **random** library for randomly generating shapes from list. Concepts of classes and objects is also included.

Many of the function like grid,time are new to us but it is understood with help of youtube,docs etc.

Reference:

Youtube channel – freeCodeCamp.org

https://www.youtube.com/watch?v=zfvxp7PgQ6c

SOURCE CODE:

```
import pygame
import random
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 blo ck
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
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:
```

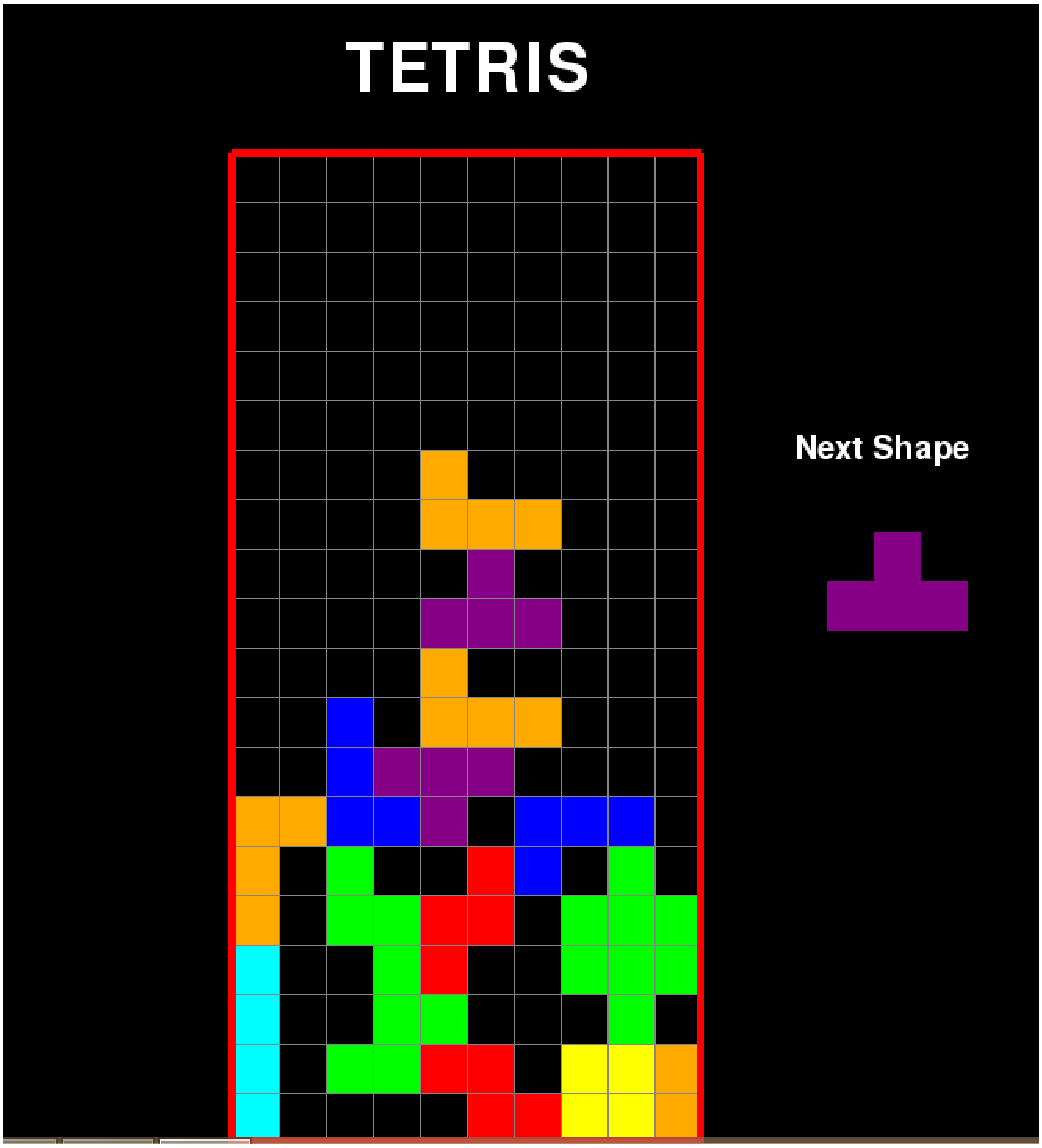
```
            c = locked_positions[(j,i)]
            grid[i][j] = c
    return grid
def convert_shape_format(shape):
    positions = []
    format = shape.shape[shape.rotation % len(shape.shape)]
    for i, line in enumerate(format):
        row = list(line)
        for j, column in enumerate(row):
            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
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))'''  # 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
            clear_rows(grid, locked_positions)
        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
```

SCREENSHORT OF OUTPUT:


TETRIS
Next Shape