TETRIS GAME

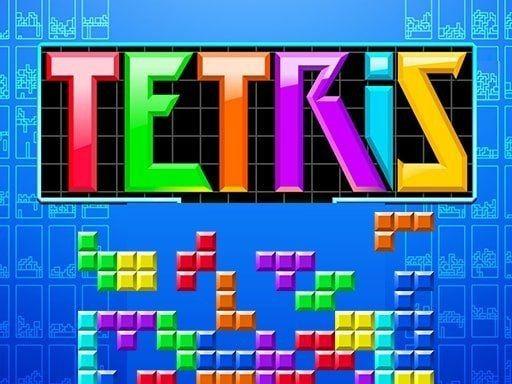
## 

## CSE 102 : TERM PROJECT

## SUPERVISOR : Khaled Mahmud Shahriar Sir

## 

## Abid Mahbub Bari 2305012 CSE A1



## OBJECTIVE:

-> To implement Tetris from scratch using OpenGL based iGraphics library.

->To demonstrate my coding, problem-solving, software design, implementation, basic graphical interface development & game mechanics development

## PROJECT OVERVIEW:

## Tetris is a classic arcade tile-matching game. Here,

## ->Any of the seven possible tetrominoes fall randomly one after another

## ->The player controls the tetrominoes by rotating, positioning it left, right or down, dropping it in place in a 20x10 grid.

## ->If any of the rows is filled with blocks, the row is cleared.

->For each cleared row, the player gets 150 points

->The game ends when the blocks cross the ceiling of the grid.

### CONTROLS:

Main Menu:

Mouse to click on “START”, “EXIT” and the “ABOUT” button

Play Pause:

Mouse to click on “RESUME”, “RESTART” and “EXIT” button

In-Game:

‘P’ to pause

‘END’ to exit from game

Game Over:

‘P’ to play again

‘M’ to return to main menu

‘END’ to exit from game

Game-Play:

‘W’ key to rotate the tetrominoes

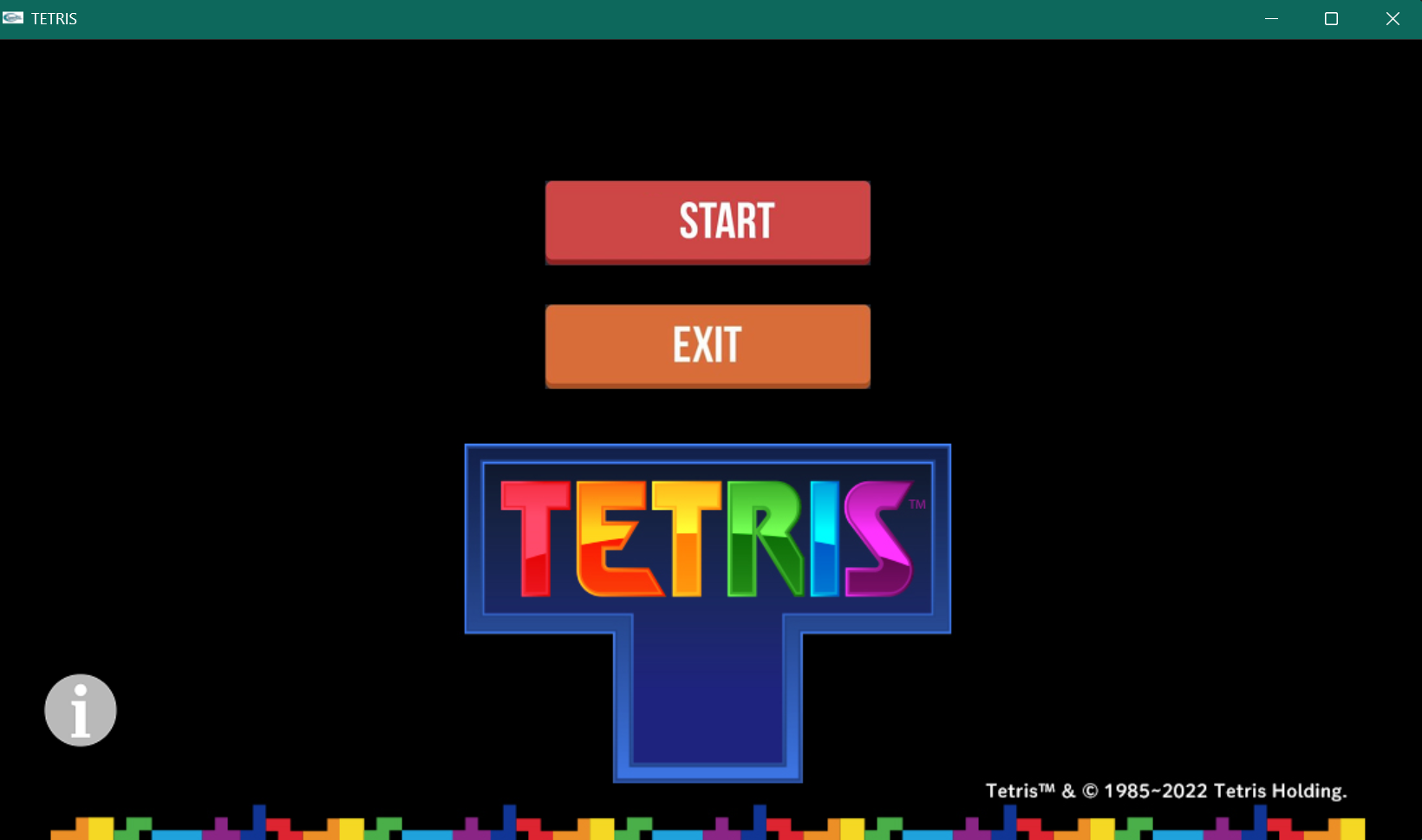
‘A’ to move the tetrominoes to the left

‘S’ to move the tetrominoes lower

‘D’ to move the tetrominoes to the right

### USER-INTERFACE:

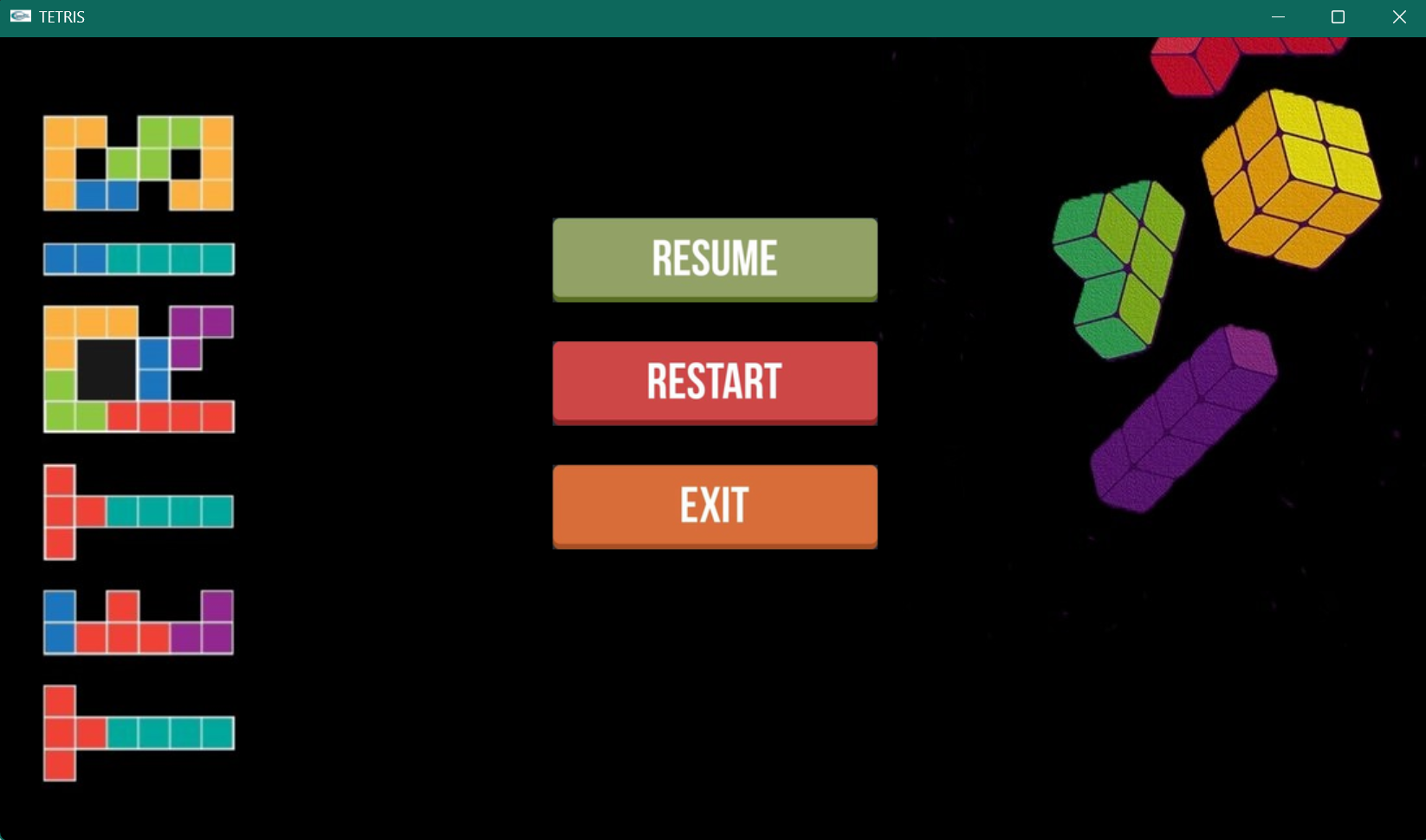
MAIN MENU:



About Section:



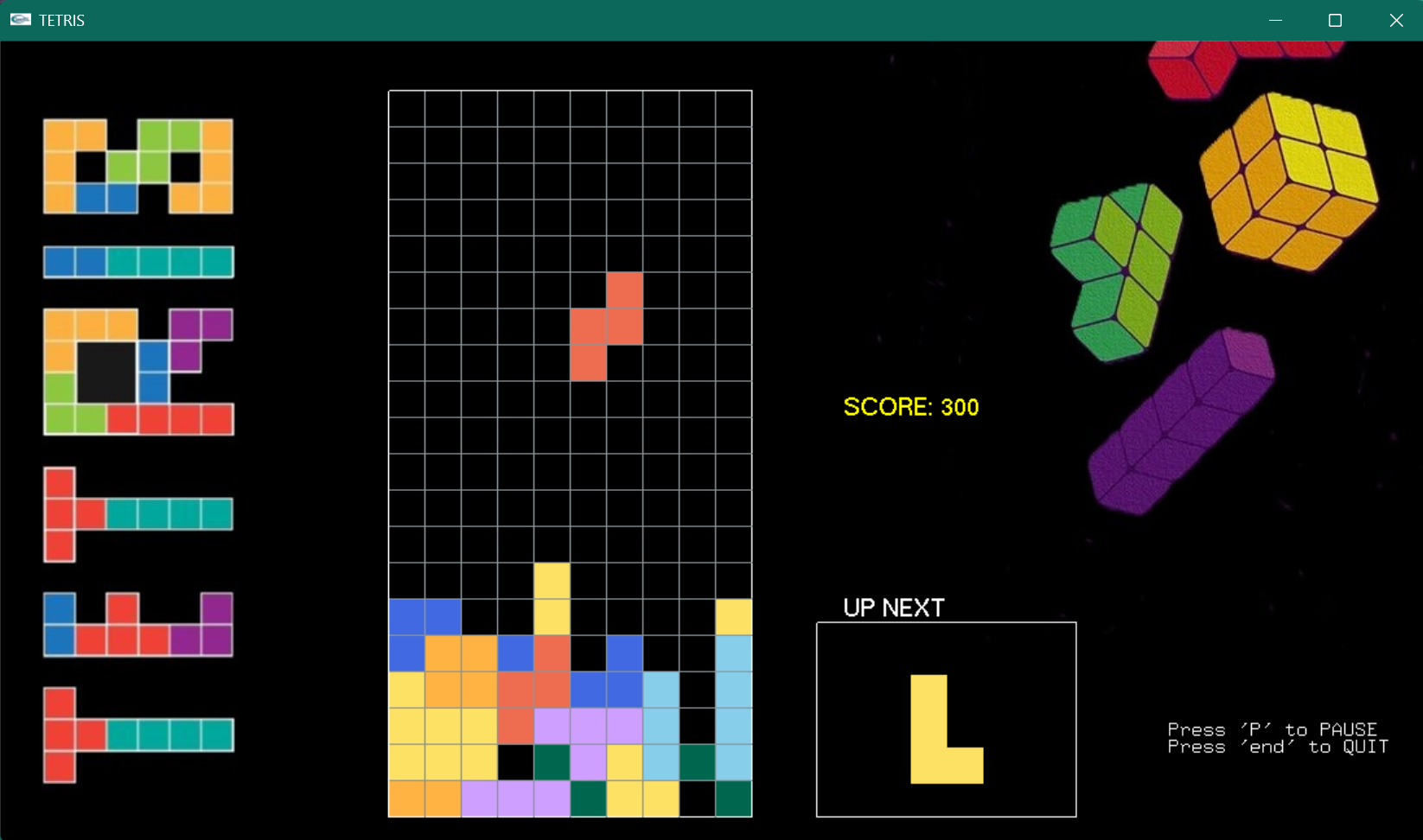
Play-Pause Menu:



Game-Over:



### GAME MECHANICS:



Rotation:

When the player clicks ‘W’ the tetrominoes are rotated 90 degrees anti-clockwise. Significant challenges were faced when implementing this which will be discussed in the “Challenges Faced” section.

Row Clearing:

The player will move the blocks according to “CONTROLS” and then place them so that the rows are filled up. When a row is filled it is cleared and the blocks above it move down by one block.

Scoring:

Player gets 150 points for each time a row is cleared.

High Score:

The highest score is stored in “data.txt” file and it is shown when the game is over.

Game Over:

The player must avoid colliding with the ceiling to keep playing and increase score.

Collision:

Collision is checked with the existing occupying blocks and the floor, left wall, right wall.

## IMPLEMENTATION DETAILS:

### Polygon/Tetromino Declaration:

### Each of the tetrominoes are defined as structures with corresponding x coordinates, y coordinates in the arrays called “base\_x[4]” and “base\_y[4]”. Each also have specific r,g,b values.

### Collision:

Collision is checked by collision\_grid\_x(), collision\_left(),collision\_right() functions which take the required coordinates of the “players\_polygon” (that the player is currently using to play) and if collision happens, it locks the blocks.

Gravity:

Gravity makes the blocks drop and the drop speed increases as the players score increases. Gravity also checks for collision, game over and high score update.

### Rotate:

Rotation is defined pixel by pixel and the rotated coordinates are stored in a temporary structure of the type “polygons”. If the rotated structure collides with the grid then rotation does not happen.

### Clear Row:

Loops through all the occupied blocks, and finds out the if a row is filled using a flag variable ‘found’. If not found it skips to the next row. If it finds a filled row, it removes those blocks and brings the blocks above it down by 1 “little\_block”.

## CHALLENGES FACED:

1. First I tried to use one of the little blocks in a tetromino as a pivot and calculate the coordinates for the others using an equation. But that failed and caused unnecessary bugs. So instead I defined the new position pixel by pixel which was tedious to implement but easy to understand and execute.
2. Clearing rows was buggy after the aforementioned function implementation. However, there was a logical error in the loops, the loops were iterating until no\_blocks for the occupied blocks (i < no\_blocks) whereas they should be iterating up to no\_blocks ( i <= no\_blocks)
3. To implement a high score, I had to learn file I/O and implement it to update the score, store it and display it.
4. All the images were converted to bmp and edited, formatted to match the game theme.
5. Music had to be converted to “wav” to support the “Windows Multimedia Library” API and use the PlaySound function.
6. The PlaySound function was throwing an error which had to be fixed by adding “-lwinmm” to the ‘arg’ in tasks.json file.
7. Some buttons could not be found online so I had to design them by myself.

……in addition to many other challenges.

## CONCLUSION:

The mentioned features were implemented, tested, debugged and after overcoming the mentioned challenges and more, a functional TETRIS game was developed to the best of my ability. Still some bugs as well as optimization issues remain.