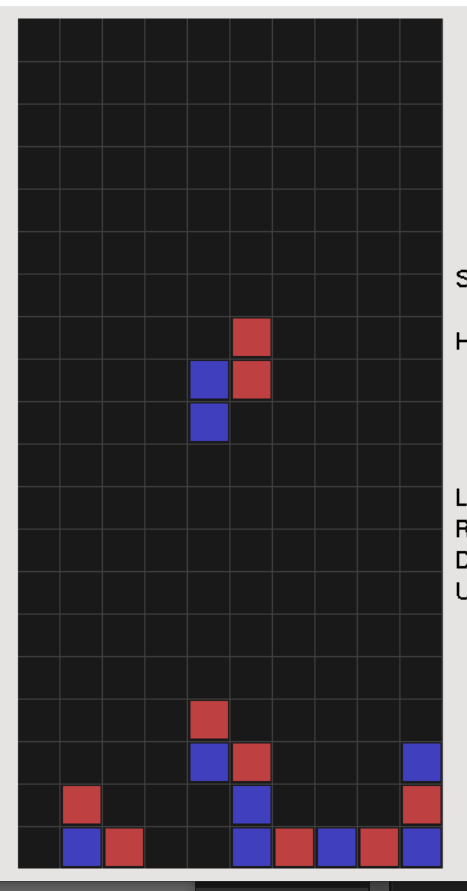
Raiyan Abrar | CSE - 19 | Student ID: 201914013

Submitted to: Lec. Oishee mam

[TETRIS]

[Project report CSE 106]



**Game Project Report: Tetris**

**Overview:** The project is a game based project created using opengl with C language as the core language. Our group mainly focused on the classic games from the older generations which consists of Tetris, Tic Tac Toe, Treasure Island and Star Wars. This project was on Tetris. The rule of the game is to try match the falling blocks each row until the blocks reach the top.

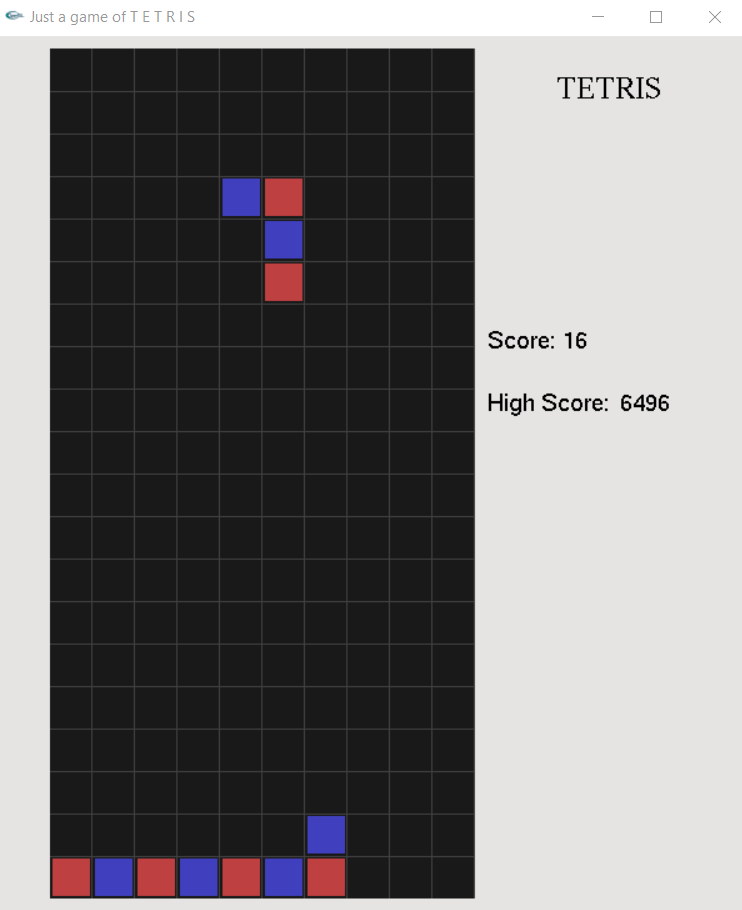
**How the project is different from other similar projects:** Each of our 4 games have the following features:

1. Graphics
2. Music
3. Interactive menu
4. Storing high scores
5. Flawless implementation

Above all, our games were combined into a single main menu from which the user can play any games he wants any number of times. For this, it can be said that our project is more user friendly which makes it different from other similar projects.

**Features of the project:** This project has some unique features. Some of the notable ones are mentioned below:

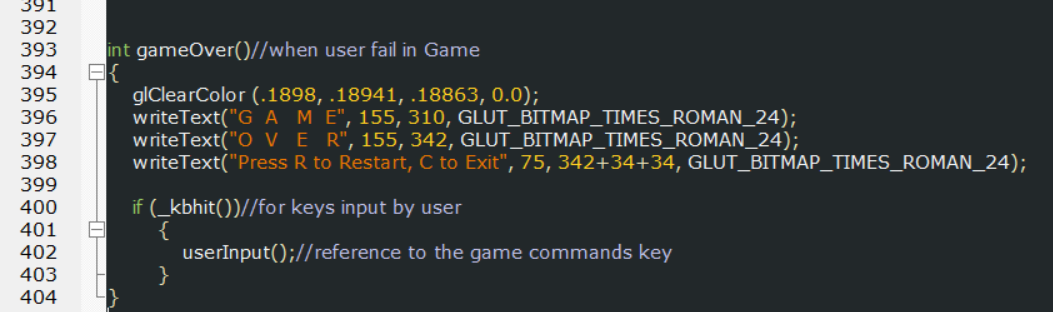
1. **Use of graphics:** The game has a user friendly graphical window which starts when user presses the ENTER key. The game starts with a 20 X 11 grid game area on the left side where one block fall each time until it reaches the bottom or on top of any other block.
2. **Interactive music:** A background music is added which plays during game time and loops every time it ends. A beep sound is also played when any row is filled with blocks and thus those blocks vanishes and respective blocks above it fall one row down.
3. **Score display:** A score board is visible on the right side of the window with current score and highscore(which is stored in a file using the FILE feature.) The score updates when any block reaches the bottom or any row is matched.
4. **Colorful blocks:** The blocks falling from the top change between red and blue throughout the game which makes the game experience much more lively adding with the music.



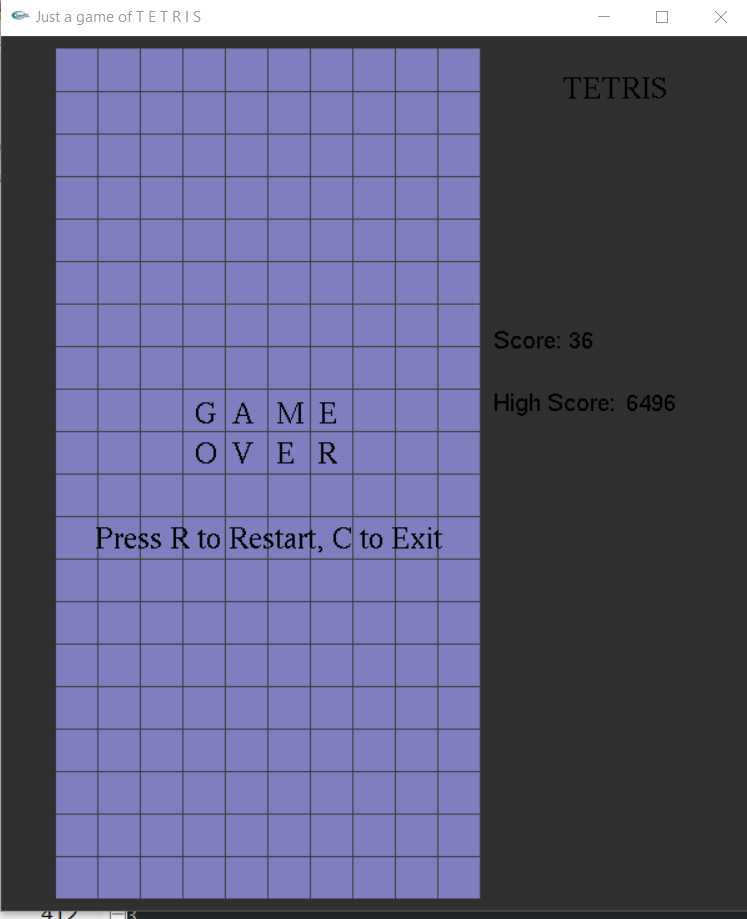
**Interaction between the components:** Different functions are added for separate purposes which together control the flow of the game. int gameOver() prints G A M E O V E R on the screen and asks the user whether he’d like to play again when the block reaches the top of the grid. void spwanBlock() checks whether it is possible to move the blocks further down through the void moveBlock(int x2, int y2) function which checks it through position x2 and y2. It checks any collision through bool isCollide(int x2, int y2) function and returns true or false. Otherwise the spwanBlock() function will just spawn a new block using the void moveBlock(int x2, int y2) function. The whole output is stored in the stage[][] array which is ultimately shown on the void display() function every function. In this way, each of the functions interact with each other through the game.

**Implementation of the game:** A brief explanation of some important games are given:

1. When the blocks reach top, gameOver() function is called:



The code first clear the game area into a lighter blue color, then it matches the grids to show game over text in separate lines. The next lines asks the user whether he’d play again. If the user presses R, new game will start, pressing C will exit the game.



1. Function for user input :

void userInput()

{

char key;

key = \_getch();

switch (key)

{

if(!gameover){

case 77:

if (!isCollide(x + 1, y))//check collision and move

{

moveBlock(x + 1, y);//check collision and move

}

break;

case 75:

if (!isCollide(x - 1, y))//check collision and move

{

moveBlock(x - 1, y);//check collision and move

}

break;

case 80:

if (!isCollide(x, y + 1))//check collision and move

{

moveBlock(x, y + 1);//check collision and move

}

break;

case ' ':

rotateBolck();//Rotation through 90 degree for every in spacekey

break;

case 72:

rotateBolck();//Rotation through 90 degree for every in spacekey

break;

}else if(gameover)

{

case 'r':

ShellExecute(NULL, "open", "Tetris project.exe", NULL, NULL, SW\_SHOWNORMAL);

exit(0);

break;

case 'R':

ShellExecute(NULL, "open", "Tetris project.exe", NULL, NULL, SW\_SHOWNORMAL);

exit(0);

break;

case 'c':

exit(0);

break;

case 'C':

exit(0);

break;

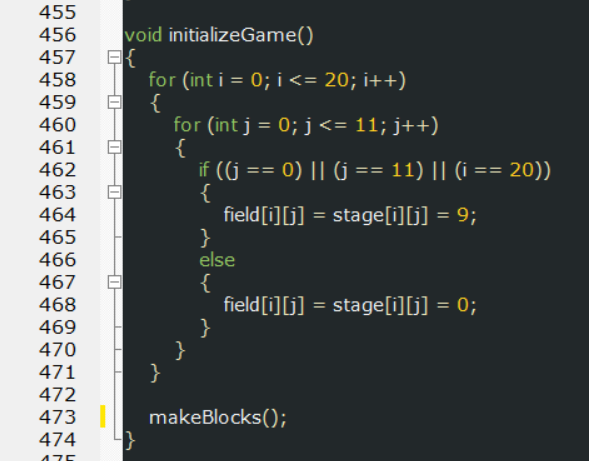
}

}

}

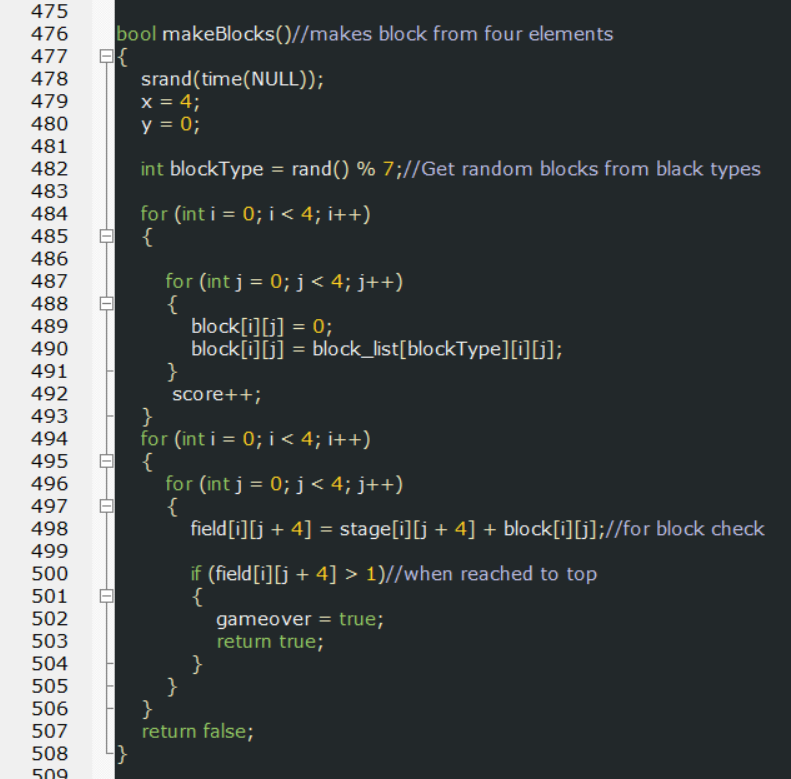
When any key is pressed, the switch() checks the input. Pressing any direction keys will make the blocks to move left, right, down or rotate only if no other block is there to block its movement. If game is over, it goes to the else part and checks whether ‘C’ or ‘R’ is pressed.

1. Initialized game:



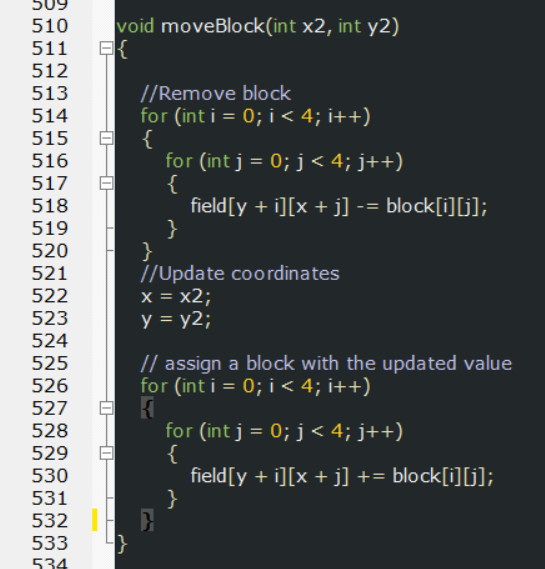
The function first initializes all the values of the stage[][] array into 0 and the sides to 9 indicating it as the boundary. Later if calls the makeBlocks() function to make the blocks.

1. Making the blocks:



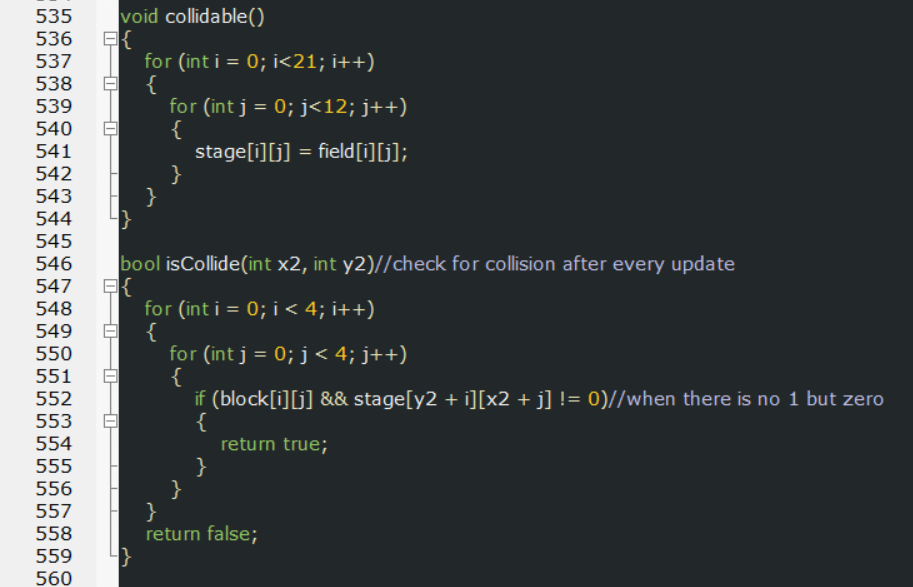
There are 7 different types of blocks in the game. blockType stores a random number between 1 to 7 to tell the game that it will spawn that block next time. The next piece of codes just draws the blocks into another array and then puts it in the stage[][].

1. Moving the blocks:



The first part of the code removes the older blocks from the stage and the next part sets the blocks to the new coordinates according the conditions. X2 and Y2 stores the new coordinates each time an update is made to the blocks.

1. Checking collisions with other blocks:



These two functions check whether the falling blocks collides with any other respective block from the stage. Returns true if collision occurs otherwise false. This collision basically tells whether the blocks should move or not.

**Difficulties faced when implementing the codes:** A number of difficulties arised when trying to implement the codes:

1. The limitations of GLUT in opengl doesn’t allow use of variations of fonts and font sizes in the code and all the fonts are in black color.
2. There is an issue of changing of color blocks every frame when the goal was to set the blocks to random colors but they get changed every frame due to limitation of GLUT.
3. The program doesn’t allow multiple audios to play parallel without the use of any extra library whose implementation is very complex. For this, the musics had to be played one at a time. Only the beep sounds can play parallel since it’s a C function.
4. There were some issues while trying to match the rows. The game was crashing. It was fixed with a simple idea obtained from a video in YouTube.

**Resources taken help from:**

1. [www.youtube.com](http://www.youtube.com)
2. <https://stackoverflow.com/>