

OOP SEM PROJECT

“TETRIS”



# Project Title:Tetris

## Group Members

## Objectives:

* The main goal is to develop tetris game using fundamental principles of oop.
* Secondly we are going to use raylib liberary for graphics and audios

## Inclusions:

* Game loop that determine how game will begin and when end
* Game mechanics to set block,change block and rotate it as well
* Graphical user interface via raylib liberary
* Audios and music
* Final result at all

## Exclusions:

* Multiplayer Functionality
* Online Play
* Detyailed customization option for user

# Key Functionalities

## Game Mechanics:

* There are I, J, S L,O,S,T,Z terminus or blocks.
* User can rotate them , change them as well.
* Shows blinking line when line completes and collide
* If user clears all lines then level increases and speed of terminus too

## Game Control:

* Allows user to control game that he can start , re start and re-sume it well.
* Use navigation keys for rotation and movement of block as well

## Game Scoring System:

* Scores are allocated on basis of lines cleared at once
* Highest scores tracked during one session of game

## Game Visualization:

* Termino visuals
* Displaying next termino or block
* Displaying Gameover screen with final score and option to restart it

## Game Audio Features:

* There is sound during rotation , clearence of line and gameover screen
* There will be a continuous sound effect with in game loop for better experience of game

## Game States:

* Start state when user starts to play the game
* Pause state to pause game
* Gameover state when terminus fill till top and user is unable to clear them

## Game techniqualities:

* It uses raylib liberary for graphics and audios
* Uses Object Oriented Pyradime(OOP) for privacy and protection of design and code
* It uses event driven programming technique to accept input that is navigation or rotation from user

# Object Oriented Programe Structure

## Game Class:

* This is the central controller of the entire game, managing game states, transitions, and main game loop activities
* Sets up the game, initializes the grid, loads the first block, and starts the background music
* Handles the loop that checks for game events, updates game states, and renders graphics
* Processes inputs for moving and rotating blocks, managing pause/resume, and game restarts
* Updates scores and adjusts game difficulty as the player progresses

## Grid Class:

* Manages the 2D grid where blocks are placed, tracking filled/empty cells and handling line clearance.
* Adds blocks to the grid and checks for collision and fitting
* Identifies and clears completed lines, adjusting the grid accordingly and updating the game score
* Draws the current state of the grid, including all placed blocks

## Block Class:

* Serves as the base class for different types of Tetris blocks (tetrominos). It defines common attributes and methods that all block types share
* Methods to move (left, right, down) and rotate the block, including collision detection to ensure moves are valid within the game grid
* Method to draw the block on the grid, which might involve translating the block’s shape and orientation into graphical representations
* Maintains the state of the block, such as current position, rotation angle, and type
* : This class likely serves as a base from which specific block types inherit. Each specific block type (I, O, T, etc.) could override some methods to customize behavior, particularly how the block rotates, given the unique rotation rules for different Tetris blocks

## Inherit Block Classes:

* L block
* I block
* J block
* T block
* Z block
* S block
* O block
* Each block type initializes its id and cells differently to represent its unique shape and rotation characteristics

## Position Class:

* A utility class designed to manage and track the position (coordinates) of blocks within the grid
* Holds x and y coordinates that define a block's position on the grid
* Assists in calculating new positions during block movements and rotations

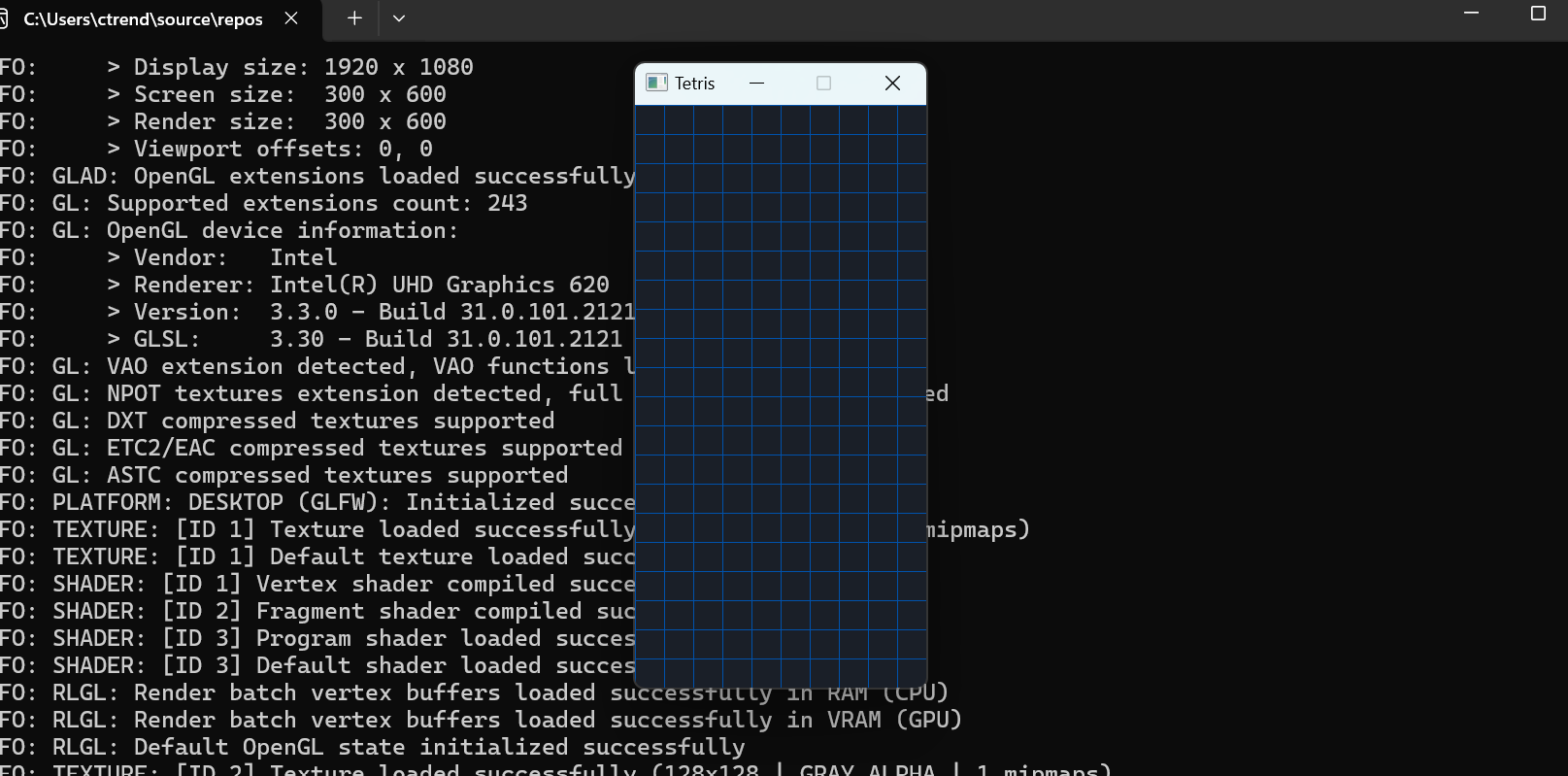
## Color Class:

* Manages the assignment of colors to different blocks, ensuring visual diversity and enhancing game aesthetics
* Defines and assigns specific colors to different types of blocks
* Provides a mechanism to retrieve the color for a particular block type, used during the rendering process

## Game Flow Using these Classes:

* The Game class initializes and controls the flow of the game, using the Grid to place blocks and clear lines
* The Blocks class (utility/factory) supplies new blocks to the game, which are then controlled by player inputs and game logic handled in the Game class
* The Block class (base) is responsible for the properties and behavior of individual blocks being played, such as their movement and rotation
* The Grid class uses functionalities from the Position and Block(base) classes to manage the placement and removal of blocks
* Visuals are managed by the Colors class, which ensures that each block is displayed with the correct color

## Graphical User Interface:

* Initial Grid:
* 
* Block:
* 
* Final Graphical User Interface:
* 