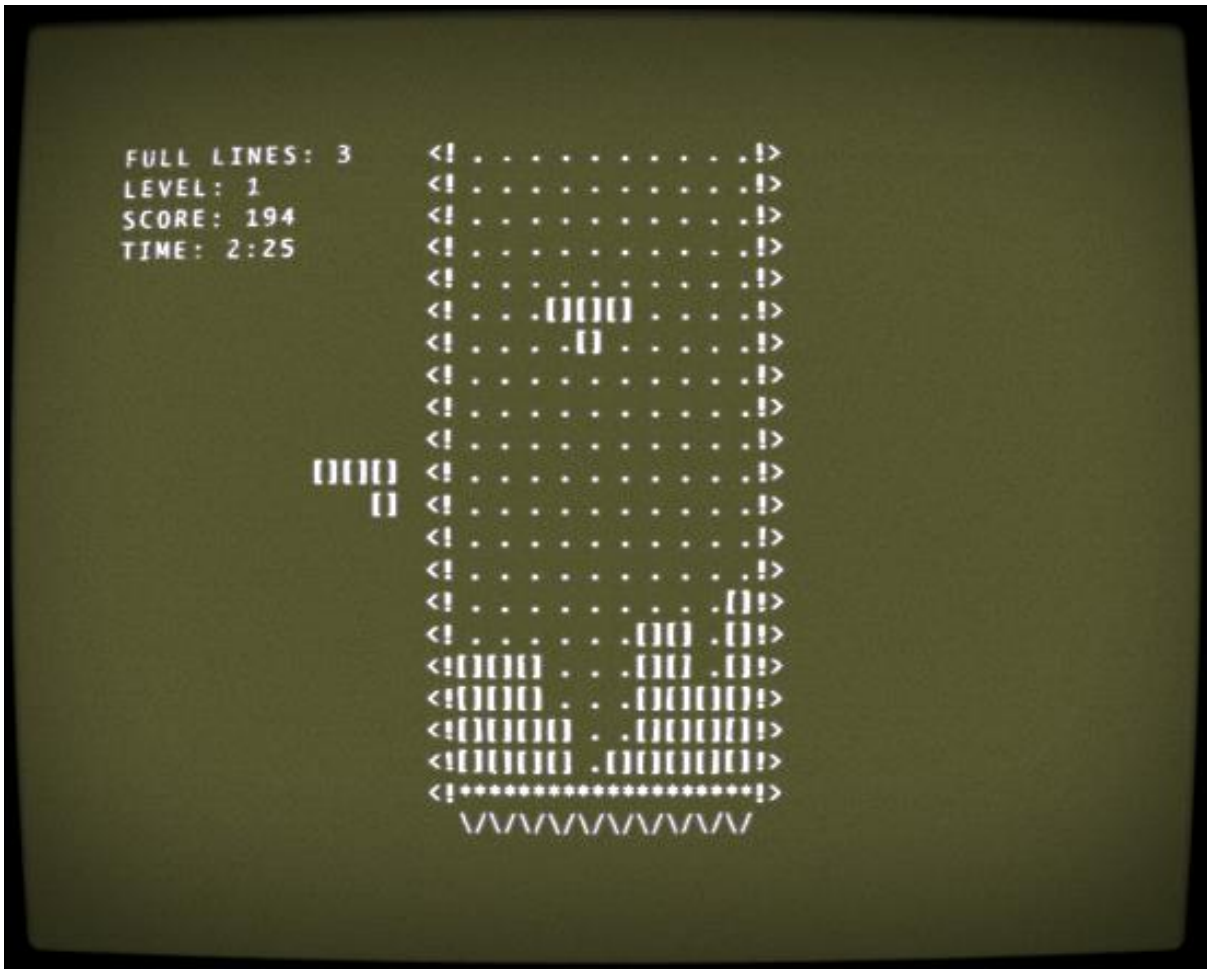
Design Overview for TetrisV

Name: Phan Vu
Student ID: 104222099

Summary of Program

I have been a fan of the Tetris game since I was just a little boy. At that time, I played it on the Gameboy console and was so into it. To be honest, I didn't think that I could recreate it. So, the program "TetrisV" is a Tetris game designed to simulate the classic gameplay experience based on what I experienced in Gameboy and the original version of 1984. It will consist of various components such as a display manager to handle screen output, a frame manager to organize game frames, controls for user interaction, a scoring system, and a Tetromino manager to handle game pieces' behavior. The game aims to allow users to play Tetris interactively, manipulating falling Tetrominos to create complete rows and achieve high scores.

A sketch of sample output to illustrate my idea.



Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: Display

Responsibility	Type Details	Notes
FrameChar	List<string>	Characters in a frame
Intro	StringBuilder	Introduction text displayed
FrameString	StringBuilder	String representation of a frame
DisplayFrame	StringBuilder	Complete frame displayed
Color	ConsoleColor	Color of characters displayed
Width	int	Width of the game screen
Height	int	Height of the game screen
Active	int	Active state of the game
Position	int	Current position in the game

Table 2: Frame

Responsibility	Type Details	Notes
Wall	Wall	Manages walls in the frame
SetIntro()	void	Sets the introduction text
SetFrame()	void	Sets the frame elements
WallList()	void	Lists all walls in the frame

*Table 3: **Wall***

Responsibility	Type Details	Notes
Values	string[]	Array of wall values
ValList	List<int>	List of wall values
Intro	bool	Boolean indicating the introduction state

*Table 4: **Key***

Responsibility	Type Details	Notes
colorKeys	Dictionary<ConsoleKey, ConsoleColor>	Key-color pairs for controls
Press()	void	Handles key press events

*Table 5: **Move***

Responsibility	Type Details	Notes
Check	Check	Manages the checking mechanism
Direction()	void	Controls the movement direction

*Table 6: **Check***

Responsibility	Type Details	Notes
Safe	bool	Indicates safe condition

Table 7: ***Preview***

Responsibility	Type Details	Notes
Next	Next	Handles the next tetromino info
Tetromino()	void	Generates the next tetromino

Table 8: ***Next***

Responsibility	Type Details	Notes
Tetromino	int	Identifier for the next tetromino
Position	int	Current position in the tetromino
Draw	List<int>	List of drawn tetromino
Random	Random	Random number generator for tetromino

Table 9: ***Reset***

Responsibility	Type Details	Notes
Now()	void	Handles the reset event

Table 10: ***Rotate***

Responsibility	Type Details	Notes
Check	Check	Manages the checking mechanism for rotation
Now()	void	Handles the rotation event
WallCheck()	bool	Checks against wall boundaries for rotation
Rotation()	void	Controls the rotation process
Inverse()	int[,]	Inverts the rotation for calculations

Table 11: ***Score***

Responsibility	Type Details	Notes
Row	Row	Manages the rows in the game
ScoreBoard	ScoreBoard	Handles the scoring information
RowCheck()	void	Checks completed rows
RowScore()	void	Calculates scores for rows
PopScoreBoard()	void	Updates the scoreboard

Table 12: ***Row***

Responsibility	Type Details	Notes
Complete	List<int>	Completed row indices
Cleared	List<int>	Cleared row indices
Counting	int	Counts completed rows
Bonus	int	Additional score bonus

Table 13: ***ScoreBoard***

Responsibility	Type Details	Notes
Rows	int	Number of completed rows
Level	int	Current level in the game
Score	int	Overall score in the game

Table 14: ***Speed***

Responsibility	Type Details	Notes
Set	Set	Manages speed settings
Check()	void	Checks speed conditions
Pause()	void	Handles pause functionality

Table 15: ***Set***

Responsibility	Type Details	Notes
Delay	int	Time delay for movements
Paused	bool	Indicates if the game is paused
Drop	bool	Controls the tetromino dropping

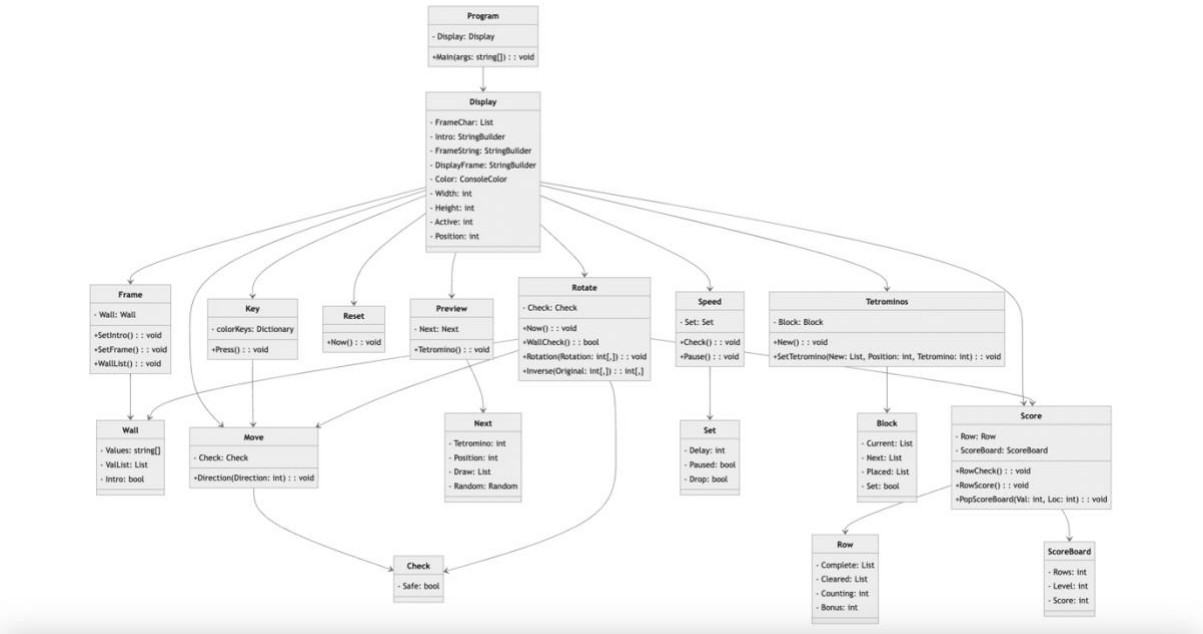
Table 16: ***Tetrominos***

Responsibility	Type Details	Notes
Block	Block	Handles block arrangements in the game
New()	void	Generates new tetrominos
SetTetromino()	void	Sets the tetromino on the board

Table 17: ***Block***

Responsibility	Type Details	Notes
Current	List<int>	Current tetromino arrangement
Next	List<int>	Next tetromino arrangement
Placed	List<int>	Placed tetromino arrangement
Set	bool	Controls the arrangement setting

Class Diagram



Sequence Diagram