Classes and Objects:

|Cell:

- A class that represents each cell of the board.

- Stores directions, cell value, and a lock status (linked list).

|Line:

- A class representing a line in the board.

- Consists of linked list nodes made of cells.

- Contains only a next pointer.

|Board:

- A class made up of multiple lines (linked list).

- Each line is a linked list of cells.

- Provides methods for operations like shrinking lines, checking and popping full lines, deleting the top line, and getting the middle element of the top line.

|Tetrimo:

- Represents each piece in the game as a 4x4 array.

- Origin is at (0, 0).

- Provides methods for changing array elements based on color and type, checking if a given position is valid, moving the piece, locking cells, deleting cells from the board, adding cells temporarily to the board, and rotating the piece.

|Next:

- A queue of tetrimos that holds items.

FUNCTIONS OF LINE

Line(int n, Cell* prev):

- Takes the previous line and generates a new one, setting it as the next of the previous line.

bool isfull():

- Checks if all cells are locked (after each piece locks)

void pop(Line* prev):

- Deletes all elements in the line and removes it from the board.

void makehead(Line* headb):

- Adds the line to the head of the board.

void shrink():

- Deletes a cell from the head of the line.

FUNCTIONS OF BOARD

Board(int a, int b):

- Creates a linked list of lines.

void shrink():

- For each line in the board, deletes the head cell.

int check_pop():

- Checks if there is a full line (is_full()), then removes it from the end and makes it the head.

void deleteTop():

- Deletes the head line.

Cell* midtop():

- Returns the middle element of the head line.

FUNCTIONS OF NEXT

Next(Cell* orjin):

- Makes three tetrimos, with the first one having the origin given by midtop().

Tetrimo* gethead():

- Returns the first element of the array.

Tetrimo* addnextone(Cell* o):

- Deletes the first element, shifts all elements one position left, and creates a new tetrimo, assigning it to the last element.

- Returns nullptr if the old one's end is false.

Tetrimo* hold(Cell* o):

- Changes the first tetrimo with end and returns the new first.

- Adds it to the board with a new origin and deletes the old first from the board.

FUNCTIONS OF TETRIMO

Tetrimo(int color, int type, Cell* o):

- Changes the array's elements with the given color value based on the given type and assigns the origin.

bool check(Cell* o, int brr[4][4]) const:

- Checks if the given origin is possible for any 4x4 array.

- Returns false if not possible.

bool move(char a):

- Checks if the move is possible using check(). If possible, moves the piece; otherwise, returns false.

- If the move is down, locks the piece.

void lockT():

- Locks each cell whose value was changed by the tetrimo.

void deleteFromBoard():

- Resets all cells changed by the tetrimo to 0.

bool addToBoard():

- Temporarily changes the board based on the array and origin.

bool rotate(int r):

- Rotates the piece, checking if it is possible. If not, checks for 3 right and 3 left rotations. Returns false if not possible.

MAIN:

- Processes user input.
  - Tab: Changes the theme based on the given index in the 3x10 matrix.
  - Esc: Exits the game.
  - B: Goes back to the main menu.
  - Up, Down, Enter: Used for selecting pages in the main menu.
  - In-game controls:
    - Down, Left, Right: Move.
    - LeftAlt (270deg), Up (90deg): Rotate.
    - P: Pause.
    - B: Back to the menu.
  - Rotations and hold are limited.
  - The game runs with the piece dropping one cell per second. After each 100 points, drop time decreases by 80%, and the board shrinks from the top and left (From 400, only from the top).

- The game calls the move function with 's' if the piece reaches the bottom, locks it, and asks for the next piece. If the next piece is null, the game is over.

ScoreBoard

After the game concludes, the system prompts the player for their name and appends it to a text file and a binary search tree. Subsequently, it displays the scoreboard utilizing various functions.

- `ScoreBoard()`: Retrieves information from a text file and constructs the binary search tree (BST).

- `void inorderTraversal(TreeNode* root)`: Modifies the `mytxt` string by traversing the BST in an inorder manner, organizing names and points separated by newline characters.

- `TreeNode* insert(std::string player, int value, TreeNode* root)`: Inserts a new element into the BST based on the associated points.

- `void save(std::ofstream& outFile, TreeNode* root)`: Persists the BST structure to the specified file.