Text-ris

Team Members:

Henry Smith • Nicholas Tateo • Benjamin Gorman • Jonathan Luzier • Dylan Maillart • James Christensen • James Maniscalco

Definition:

The system will be a User-facing, Tetris-inspired, arcade-style game called "Text-ris". The user controls blocks of various shapes as they "fall" towards the bottom of the game board and must stack them as they reach the bottom. The User's objective is to completely fill rows of the game board with blocks; when a row is completely filled, it is cleared and all the blocks in that row are removed. To keep the game going, the User must keep the blocks from stacking all the way to the top of the screen by clearing rows faster than the blocks can accumulate. The goal of this system is to entertain the User.

Analysis:

The input will consist of the User's interaction with the game via menu selections and use of the in-game controls. These controls and menu selections will be keyboard-based and restricted to certain predetermined keys; any extraneous key input by the user will be ignored. The program will accept the user's key entries as `char` data.

The output will be the graphics printed to the terminal window. During gameplay, the User's manipulation of the keyboard will control the position of the falling

block and rotate its orientation; within menus or help screens, the User will be able to make selections using predetermined keys.

The proposed system will need to track the position of the falling block and update its course upon User input; it will simultaneously need to determine when and where the falling block collides with the edges of the game board or blocks that have already landed and then "place" the falling block in a fixed position on the game board. The program will need to "know" the dimensions of the game board and will need to update the screen each time a change is made in the game (such as the position of the falling block). It will also be able to perpetually create new falling blocks for the game to continue and simultaneously manage the resources of falling blocks that have already landed.

Design:

We will be using three modules: the GameBoard module, the FallingBlock module, and the Controller module. The Gameboard module will include the Gameboard class, which keeps track of the board and the filled coordinates on the board. It will have an updateBoard() function, a translate() function, and a collision() function, along with accessor methods for the private data (coordinates of the fallen blocks). The FallingBlock module will include a FallingBlock class, which keeps track of the current block that is falling and being controlled by the user. There will then be seven child classes of the FallingBlock class, one for each individual type of block (L-shape, square, etc.). The FallingBlock class will have a rotate() method and accessor methods for the private data (coordinates of the block). The final module is the Controller module, which includes the Controller

class. It will have a menu() function, a pause() function, a restart() function, and a quit() function. There are no shared classes between the three modules.

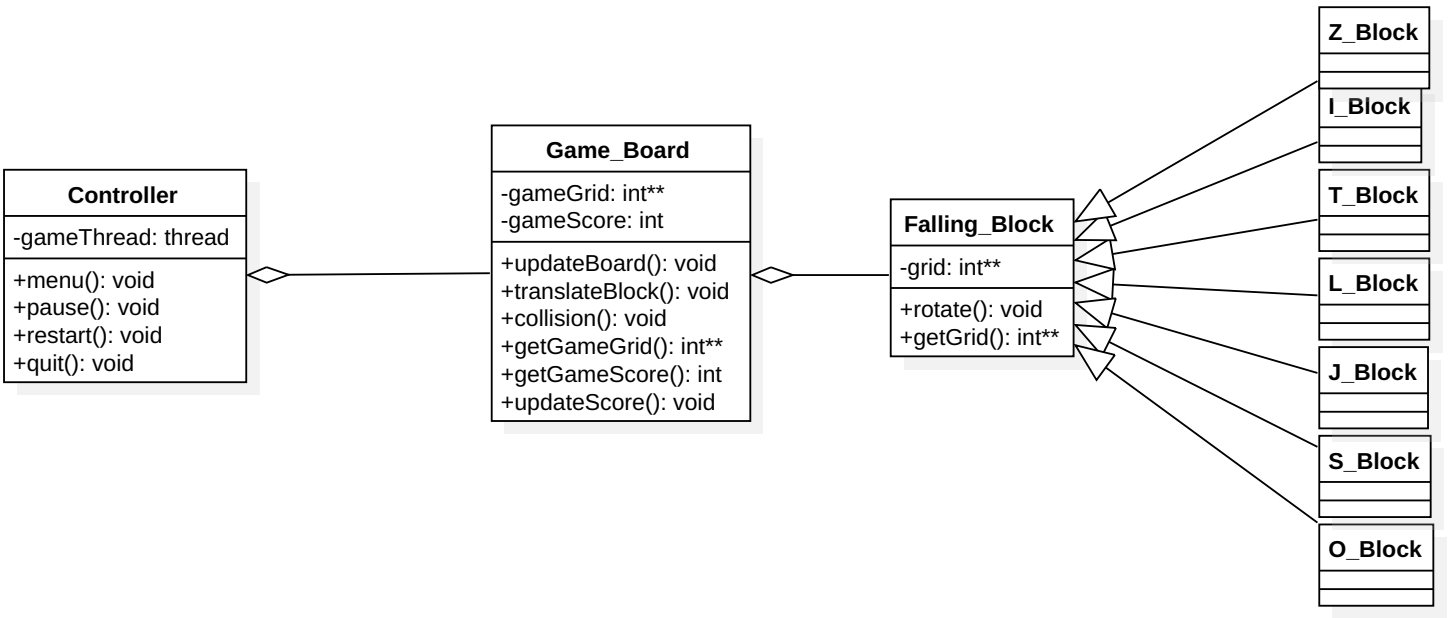
Execution Plan:

Our members will be split into three groups, each of which will work on one module of the program (Blocks, Game Board, and Controller).

Phase	Date
GameBoard Implementation	Week 1
FallingBlock Implementation	Week 1 & 2
GameBoard Testing	Week 2
FallingBlock Testing	Week 3
Controller Implementation	Week 2 & 3
Controller & Overall Testing	Week 4
“Phase 7” (Final debugging and enhancements)	Week 5

The makefile will look like:

```
make: GameBoard.cpp FallingBlock.cpp Controller.cpp
g++ -o make GameBoard.cpp FallingBlock.cpp Controller.cpp
```


Controller
-gameThread: thread
+menu(): void
+pause(): void
+restart(): void
+quit(): void
Game_Board
-gameGrid: int**
-gameScore: int
+updateBoard(): void
+translateBlock(): void
+collision(): void
+getGameGrid(): int**
+getGameScore(): int
+updateScore(): void
Falling_Block
-grid: int**
+rotate(): void
+getGrid(): int**
Z_Block
I_Block
T_Block
L_Block
J_Block
S_Block
O_Block