**Tetris Plan Overview**

1. Build Tetris Grid
   1. Array of 20 rows and 10 columns
2. Build and Draw Tetris Blocks
   1. J Shape
   2. L Shape
   3. O Shape
   4. I Shape
   5. T Shape
   6. S Shape
   7. Z Shape
3. Allow Blocks to Move left right down, but not out of Grid
4. Allow Blocks to Rotate, but not out of grid
5. Spawn Random Blocks
6. Lock Blocks when they reach bottom, Create a new Block
7. Prevent Blocks from stacking on the same cell of a grid
8. Make Blocks automatically move down over time
9. Game Over when blocks reach top
10. Clear Rows when a row is filled
11. Build Full Window and Interfaces
    1. Include Grid into Window
    2. Add Score
    3. Add Next Block diagram
    4. Add Game Over Text when game over
    5. Sounds
12. Improvements
    1. Difficulty
    2. Press Space bar to drop
    3. Change background color
    4. Randomize spawn rotation
    5. TBA

**Step 1) Build Tetris Grid**

1. Set up Window
2. Create Constants to set Cell size, Grid width, and Grid height
3. Create Constants for rows and columns of the grid
4. Create a Grid Class
5. Draw the Grid onto the Window

Grid[20][10]

Draw total of 400 cells

**step 2) Build Tetris Blocks**

Build Parent Block

1. Build Child Block for J, L, I, O, T, S, Z Blocks

Block will hold 4 values called Positions

For example: Position ={0,1}, {1,0}, {1,1}, {1,2} for one of the T blocks

Hence, Position is made of two values of Rows and Columns

Block will hold 4 values called Positions

Ex. Block 1 = Pos 1, Pos 2, Pos 3, Pos 4

Each Position will hold 2 values of a cell

Ex. Pos 1 = int x, int y

Class Block is std::map<int, std::vector<Position>> block  
Class Position is int x, int y

Drawing Blocks will require

1. Block rotation (contain map of Positions)
2. Positions based on rotation number 0 to 3 (need to get vector of Positions from block)
3. Color

J Blocks

|  |  |  |
| --- | --- | --- |
|  |  | X |
|  |  | X |
|  | X | X |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| X |  |  |
| X | X | X |

|  |  |  |
| --- | --- | --- |
| X | X |  |
| X |  |  |
| X |  |  |

|  |  |  |
| --- | --- | --- |
| X | X | X |
|  |  | X |
|  |  |  |

L Blocks

|  |  |  |
| --- | --- | --- |
| X |  |  |
| X |  |  |
| X | X |  |

|  |  |  |
| --- | --- | --- |
| X | X | X |
| X |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | X | X |
|  |  | X |
|  |  | X |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | X |
| X | X | X |

I Blocks

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| X | X | X | X |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | X |  |
|  |  | X |  |
|  |  | X |  |
|  |  | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| X | X | X | X |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | X |  |  |
|  | X |  |  |
|  | X |  |  |
|  | X |  |  |

O Blocks

|  |  |  |
| --- | --- | --- |
| X | X |  |
| X | X |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| X | X |  |
| X | X |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| X | X |  |
| X | X |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| X | X |  |
| X | X |  |
|  |  |  |

T Block

|  |  |  |
| --- | --- | --- |
|  | X |  |
| X | X | X |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | X |  |
|  | X | X |
|  | X |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| X | X | X |
|  | X |  |

|  |  |  |
| --- | --- | --- |
|  | X |  |
| X | X |  |
|  | X |  |

S Block

|  |  |  |
| --- | --- | --- |
|  | X | X |
| X | X |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | X |  |
|  | X | X |
|  |  | X |

|  |  |  |
| --- | --- | --- |
|  | X | X |
| X | X |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | X |  |
|  | X | X |
|  |  | X |

Z Block

|  |  |  |
| --- | --- | --- |
| X | X |  |
|  | X | X |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | X |
|  | X | X |
|  | X |  |

|  |  |  |
| --- | --- | --- |
| X | X |  |
|  | X | X |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | X |
|  | X | X |
|  | X |  |

Test if Drawn in main.cpp by manually instantiating one of the Block Class

**Step 3) Allow Blocks to Move left right down, but not out of Grid**

1. Have Blocks spawn in the middle
2. Allow Blocks to be manually moved Left, Right, Down
3. Prevent Block from going out of the grid

Move Block Function

1. Add a rowOffset and columnOffset variable
2. Every time a position is needed, create a new vector of positions, with each cell adjusted by the Offset, and return that vector of positions
   1. Function GetPositions()
   2. This will create a copy of positions, adjusted by offset
3. The Draw function will draw the new locations

Key Pressed Function

1. When Right, Left or Down is selected, it will move the Block from the Move Block Function
2. Use IsKeyPressed raylib function in main
   1. Key right columnOffset + 1
   2. Key left columnOffset –1
   3. Key down rowOffset +1
3. Need something to indicate block is only W wide or H high
   1. Row limit for Z block is 18, but I block is 16.
   2. Check if each positions of each rotation is out of grid

Test Blocks move in main.cpp by manually calling the MoveBlockKeyPress() function

**Step 4) Allow Blocks to Rotate, but not out of grid**

1. Pressing KEY\_UP will set the rotation to next state
2. Rotation have four state from 0 to 3, with 4 Position each as see earlier in Block <map>
3. Prevent rotation is Block ends up out of grid

Create RotateBlock Function

**Step 5) Spawn Random Blocks**

1. Instead of manually drawing a block in main.cpp, spawn one of the random blocks
2. Also have a next random block available

Create a private currentBlock and nextBlock instance of a class Block inside of new Game class.

Also create a mainGrid isntance of a class Grid inside the Game class.

Create a CreateBlock() Function, which randomly creates one of the 7 blocks. Give it to currentBlock

**Step 6) Lock Blocks when they reach bottom, Create a new Block**

1. Check if Block have reached bottom
   1. Block have reached bottom, if the IsBlockOutOfGrid() returns true for MoveDown() fucntion
2. If the value of the grid[ROWSIZE][COLUMNSIZE] is 0, paste Block drawing onto grid changing the grid value to block id

Moved all of Block manuever funtions from Block.cpp to Game.cpp. That way I can lock Block onto the Grid through game.cpp. Added a LockBlock() function to game.cpp which is called inside of MoveDown() function.

Assigned nextBlock to currentBlock, call CreateBlock() to nextBlock.

**Step 7) Prevent Blocks from stacking on the same cell of a grid**

1. Check the value of Grid.
2. If the grid is anything but 0 during MoveDown(), it will Lock the Block.
3. If the grid is anything but 0 during MoveLeft or MoveRight, it will prevent movement
4. If the grid is anything but 0 during Rotate, it will prevent rotation

**Step 8) Make Blocks automatically move down over time**

1. Count number of frames that have passed
2. If number of frames past a chosen threadshold, call the MoveDown() function

**Step 9) Game Over when blocks reach top**

1. Create a bool variable called GameOver
2. Only allow the controls (Rotate, MoveDown, MoveLeft, MoveRight) when GameOver is false
3. In LockBlock() function, call IsGridCellFilled() function the moment after currentBlock is assigned nextBlock
4. If the grid is filled the moment the block spawn, it means block is at top
5. Set GameOver to true
6. Allow Game to restart when Enter is pressed during this state
   1. Clear all blocks and set all grid back to 0

**Step 10) Clear Rows when a row is filled**

1. After LockBlock(), check if each row, starting from 20 counting down towards 0 is full.
2. If detect a full row, check if it’s a single row or 2, 3, 4 rows in sequence
3. Adjust row

**Step 11)** Build Full Window and Interfaces

1. Include Grid into Window
   1. Create Window Constant: Height GRID\_HEIGHT + 20, width GRID\_WEIGHT \* 1.6
   2. Replace this size in InitWindow
   3. Place Grid inside of Game
2. Add Score
   1. Draw Scoreboard Diagram
   2. Draw score text
3. Add Next Block diagram
   1. Draw NextBlock Diagram
   2. Draw the nextblock in there
4. Add Game Over Text when game over
5. Sounds
   1. TBA

**Step 12) Improvements**

* 1. Difficulty - Added
  2. Press Space bar to drop - Added
  3. Change background color
  4. Randomize spawn rotation
  5. TBA

**Valuable Lesson**

1. Learned how to use <maps>
2. Learned to check if the return value is correct
   1. In a function I built that returns a <vector>, I was trying to return the <vector> class itself, instead of returning a vector variable with stored data
3. Instead of being fancy trying to pass arguments from function to function, simplify by creating new variables within the class.
   1. When I tried to move blocks, I would create a new Block, new Position, and pass them function to function. However, by creating rowOffset and columnOffset int variable, I only needed to add few new lines in the GetPosition() function.
   2. Before using this method, I would try to change the position within the child Block class itself, but this is undesirable as we want to keep the same default position every time
4. Keep order of constants consistent
   1. Sometimes I would have (column, row) and others (row, column)
5. Learned to reuse a Function for other purposes
   1. I was using GetPosition() function to grab the positions of a block to Draw them. This function returned a new vector with int row and int column updated to their respective position.
   2. However outside of the Draw function, I was manually adjusting for positions, when I could of just reused the GetPosition() function.
6. When Testing a Function that has a return value, ensure to have the function return a value for ALL conditions.
   1. I was trying to test case 1 out of 7. If the game try to return 2-6 without those return being set, the compiler will complain.