Period 2
Group Members: Ivan Li and Brandon Chen
Group Name: Budget 99
Project Title: Tetris 99

Brief Project Description: This project aims to emulate Tetris, in which blocks known as tetronimones slowly fall. The goal of the game is to either achieve some personal best score for singleplayer or to last longer than your opponent in multiplayer. This is done through a simple mechanic in which every time a row is completely filled, it's cleared. You will gain some corresponding score based on how many rows are completely filled from the current action in single player or send some amount of "garbage" to the opponent which will fill up the opponent's grid.

List of Current Functionalities:

- Tetrominoes (Tetris blocks are moved by the WASD and arrow keys)
- Background grid (Singleplayer and multiplayer)
- Standard gravity (Tetris blocks fall down on their own)
- Borders (Blocks cannot move out of the playing zone)
- The tetrominoes actually gets put onto the grid, and stays until a row is cleared (Update the background simultaneously with the tetrominoes)
- Score Counter (When a row is cleared, add to current score)
    - Highest Score (Shows the highest score you obtained in this current session in single player)
- Next Block Screen (Indicates the next blocks that will drop)
- End Screen (Different Text Message appears based on who wins)

List of Functionalities to be done by the deadline:

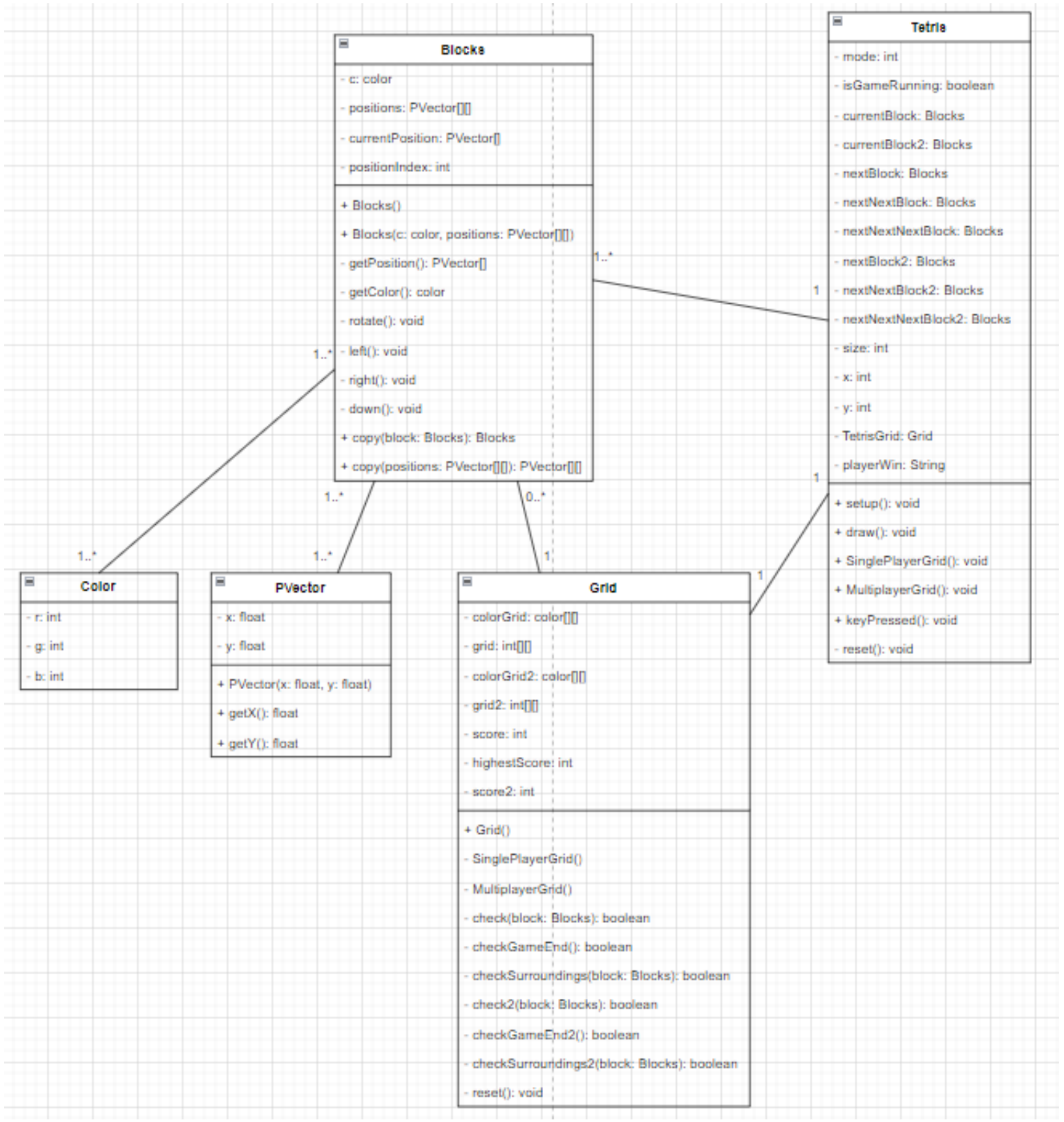
- Quick drop (Tetris blocks go all the way down to the bottom instead of having to press the down key multiple times in a row)
- Garbage (When you clear a row in multiplayer, send a row of "garbage" to the other side)
- Drop Indicator (Shows where the block will be placed if you were to quick drop)
- Naïve gravity (Tetris blocks fall down on their own, and the pace increases as the game goes on)

Troubleshooting:

- We had trouble with making the tetrominoes stay on the grid, and having a full row clear. We fixed this issue by having two 2-dimensional arrays, called grid and colorGrid. The 2-D array grid keeps track of whether there is a block at a specific location or not. If a position in the grid is 0, it means that there is no block there. If a position in the grid is 1, it means that there is a block there. The color grid corresponds accordingly with the grid,

storing the colors that the blocks are. This allowed us to keep track of when there was a full row and to keep the blocks on the grid with their corresponding colors.

- The next block screen was difficult at first, however, we realized that it would be easy to implement if we create more block objects that can represent the “next” blocks. This allowed us to store the “next” blocks and display them on screen for the player to see, and set the references to a block to the next block when a block reaches the bottom.
- The borders were implemented by making sure that the blocks cannot go out of the playing area using the coordinates of the grid.
- The end screen simply uses the text function that processing has.


Blocks
- c: color
- positions: PVector[][]
- currentPosition: PVector[]
- positionIndex: int
+ Blocks()
+ Blocks(c: color, positions: PVector[][])
- getPosition(): PVector[]
- getColor(): color
- rotate(): void
- left(): void
- right(): void
- down(): void
+ copy(block: Blocks): Blocks
+ copy(positions: PVector[][]): PVector[][]
Tetris
- mode: int
- isGameRunning: boolean
- currentBlock: Blocks
- currentBlock2: Blocks
- nextBlock: Blocks
- nextNextBlock: Blocks
- nextNextNextBlock: Blocks
- nextBlock2: Blocks
- nextNextBlock2: Blocks
- nextNextNextBlock2: Blocks
- size: int
- x: int
- y: int
- TetrisGrid: Grid
- playerWin: String
+ setup(): void
+ draw(): void
+ SinglePlayerGrid(): void
+ MultiplayerGrid(): void
+ keyPressed(): void
- reset(): void
Color
- r: int
- g: int
- b: int
PVector
- x: float
- y: float
+ PVector(x: float, y: float)
+ getX(): float
+ getY(): float
Grid
- colorGrid: color[][]
- grid: int[][]
- colorGrid2: color[][]
- grid2: int[][]
- score: int
- highestScore: int
- score2: int
+ Grid()
- SinglePlayerGrid()
- MultiplayerGrid()
- check(block: Blocks): boolean
- checkGameEnd(): boolean
- checkSurroundings(block: Blocks): boolean
- check2(block: Blocks): boolean
- checkGameEnd2(): boolean
- checkSurroundings2(block: Blocks): boolean
- reset(): void
1..*
1
0..*