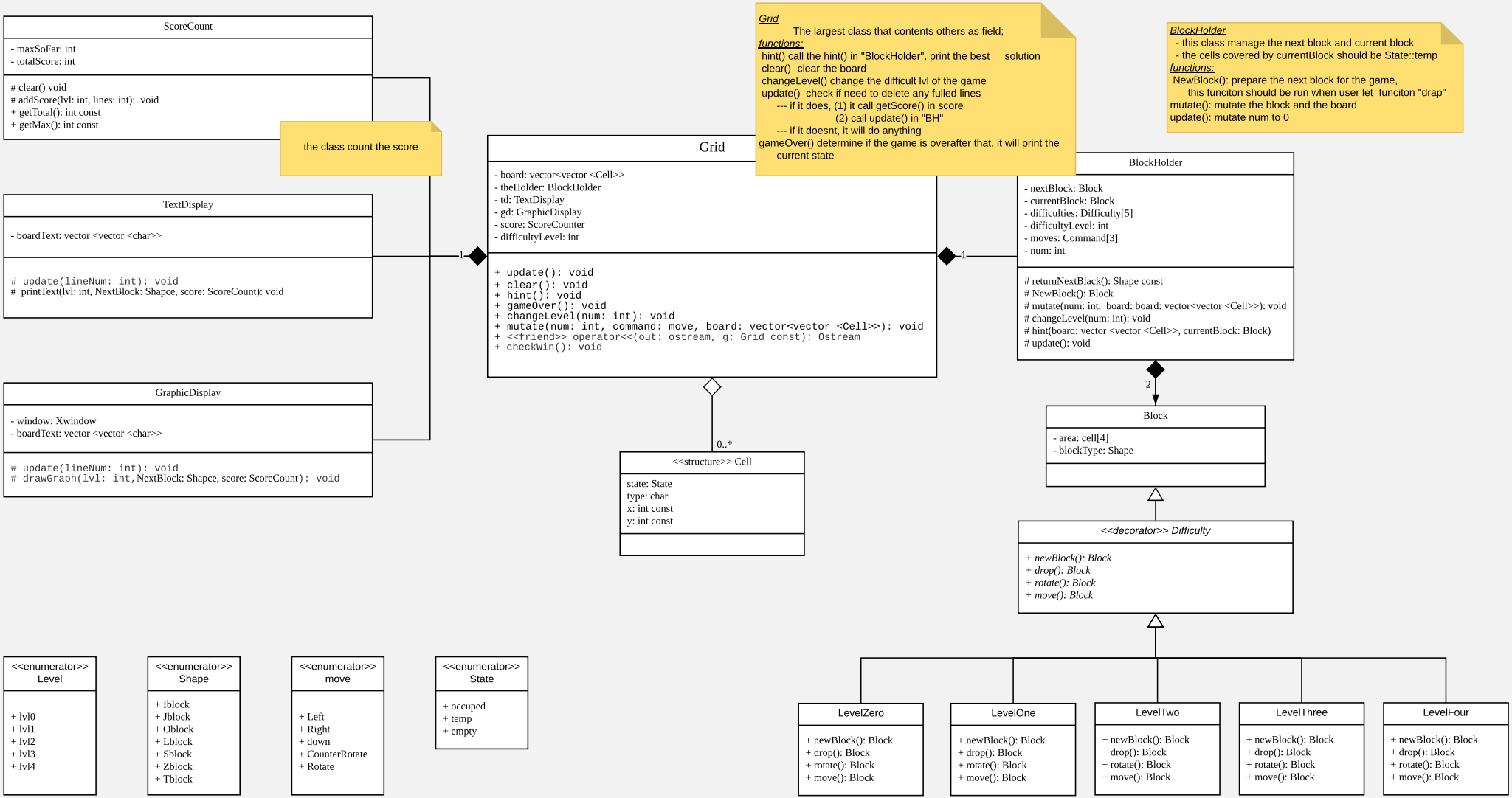
ScoreCount
- maxSoFar: int
- totalScore: int
clear() void
addScore(lvl: int, lines: int): void
+ getTotal(): int const
+ getMax(): int const
the class count the score
TextDisplay
- boardText: vector <vector <char>>
update(lineNum: int): void
printText(lvl: int, NextBlock: Shapce, score: ScoreCount): void
GraphicDisplay
- window: Xwindow
- boardText: vector <vector <char>>
update(lineNum: int): void
drawGraph(lvl: int, NextBlock: Shapce, score: ScoreCount): void
Grid
- board: vector<vector <Cell>>
- theHolder: BlockHolder
- td: TextDisplay
- gd: GraphicDisplay
- score: ScoreCounter
- difficultyLevel: int
+ update(): void
+ clear(): void
+ hint(): void
+ gameOver(): void
+ changeLevel(num: int): void
+ mutate(num: int, command: move, board: vector<vector <Cell>>): void
+ <<friend>> operator<<(out: ostream, g: Grid const): Ostream
+ checkWin(): void
Grid
The largest class that contents others as field;
functions:
hint() call the hint() in "BlockHolder", print the best solution
clear() clear the board
changeLevel() change the difficult lvl of the game
update() check if need to delete any fulled lines
--- if it does, (1) it call getScore() in score
(2) call update() in "BH"
--- if it doesnt, it will do anything
gameOver() determine if the game is overafter that, it will print the current state
BlockHolder
- nextBlock: Block
- currentBlock: Block
- difficulties: Difficulty[5]
- difficultyLevel: int
- moves: Command[3]
- num: int
returnNextBlack(): Shape const
NewBlock(): Block
mutate(num: int, board: board: vector<vector <Cell>>): void
changeLevel(num: int): void
hint(board: vector <vector <Cell>>, currentBlock: Block)
update(): void
BlockHolder
- nextBlock: Block
- currentBlock: Block
- the cells covered by currentBlock should be State::temp
functions:
NewBlock(): prepare the next block for the game,
this funciton should be run when user let funciton "drap"
mutate(): mutate the block and the board
update(): mutate num to 0
1
1
0..*
2
<<structure>> Cell
state: State
type: char
x: int const
y: int const
Block
- area: cell[4]
- blockType: Shape
<<decorator>> Difficulty
+ newBlock(): Block
+ drop(): Block
+ rotate(): Block
+ move(): Block
<<enumerator>> Level
+ lvl0
+ lvl1
+ lvl2
+ lvl3
+ lvl4
<<enumerator>> Shape
+ Iblock
+ Jblock
+ Oblock
+ Lblock
+ Sblock
+ Zblock
+ Tblock
<<enumerator>> move
+ Left
+ Right
+ down
+ CounterRotate
+ Rotate
<<enumerator>> State
+ occuped
+ temp
+ empty
LevelZero
+ newBlock(): Block
+ drop(): Block
+ rotate(): Block
+ move(): Block
LevelOne
+ newBlock(): Block
+ drop(): Block
+ rotate(): Block
+ move(): Block
LevelTwo
+ newBlock(): Block
+ drop(): Block
+ rotate(): Block
+ move(): Block
LevelThree
+ newBlock(): Block
+ drop(): Block
+ rotate(): Block
+ move(): Block
LevelFour
+ newBlock(): Block
+ drop(): Block
+ rotate(): Block
+ move(): Block