App

+ CELLSIZE: int = 32
+ TOPBAR: int = 64
+ WIDTH: int
+ HEIGHT: int
+ FPS: int = 30
+ configPath: String
+ random: Random
+ NUM_ROWS: int = (HEIGHT - TOPBAR) / CELLSIZE
+ NUM_COLUMNS: int = WIDTH/CELLSIZE
+ level: Level
+ colors: String[] = {"grey", "orange", "blue", "green", "yellow"}
+ colorsIndices: String[] = {"0", "1", "2", "3", "4"}
+ colorsIndices: int[]= {0, 1, 2, 3, 4}
+ score_increase: Map<String, Integer>
+ score_decrease: Map<String, Integer>
+ board: Block[][]
+ balls_image: Map<String, PImage>
+ tile_image: PImage
+ entrypoint: PImage
+ walls_image: Map<String, PImage>
+ break_walls_image: Map<String, PImage>
+ holes_image: Map<String, PImage>

+ settings(): void
+ setup(): boolean
+ keyPressed(event: KeyEvent): void
+ mousePressed(e: MouseEvent): void
+ mouseDragged(e: MouseEvent): void
+ rightDraggedHelper(): void
+ mouseReleased(e: MouseEvent): void
+ draw(): void
+ drawWinningAnimation(): void
+ drawTopBar(): void
+ drawMap(): void
+ drawBallsOnTheMap(): void
+ drawTimer(): void
+ drawScore(): void
+ drawBallTimer(): void
+ drawNextFiveBalls(): void
+ drawLines(): void

Level

+ currentLevel: int
+ levels: JSONArray
+ characters: String[ ]= {"X","S","H","B"," ","0","1","2","3","4"}
+ layoutFile: String
+ layoutContent: ArrayList<ArrayList<String>>
+ time: int
+ remainTime: float
+ spawn_interval: int
+ spawner_number: int
+ increaseModifier: float
+ decreaseModifier: float
+ ballsQueue: ArrayList<Ball>
+ ballsOnTheMap: ArrayList<Ball>
+ firstFrame: boolean
+ resetScore: float
+ currentScore: float
+ linesCollection: ArrayList<Line>
+ currentline: Line
+ levelStatus: String
+ frameElapsedForBallTimer: int
+ frameElapsedForTimer: int
+ frameForWiningAnimation: int

+ enterNextlevel(): void
+ hasNextlevel(): boolean
+ updateMapSetting(Score: float): void
+ readLayoutFile(): void
+ setupBoard(): void
+ addBallToTheMap(newBall: Ball): void
+ addLine(): void
+ deleteLine(): void
+ serveBall(): void

Ball

+ color: String
+ colorIndex: String
+ x: float
+ y: float
+ radius: float
+ x_velocity: float
+ y_velocity: float
+ ready: boolean

+ updateVelocity(xv: float, yv: float): void
+ updateColor(color: String): void
+ CheckEnterHole(level: Level,a: App): Ball
+ checkCollision(app: App): void
+ boundaryCollisionCheck(): void
+ wallCollisionCheck(): void
+ lineCollisionCheck(App app): void
+ checkCollisionHelper(p1x: float, p1y: float, p2x: float, p2y: float, episilon: float): boolean
+ checkLineCollisionHelper(p1x: float, p1y: float, p2x: float, p2y: float): boolean
+ normalVectorHelper(p1x: float, p1y: float, p2x: float, p2y: float): float[ ]
+ ballMove(): void

Block

+ x: int
+ y: int
+ animation: boolean
+ blockType: String
+ ImageBlock: boolean
+ imageBlockSize: int
+ color: String
+ colorIndex: String
+ bouncy: boolean
+ bouncyNum: int
+ upperLeft: float[ ]
+ upperRight: float[ ]
+ bottomLeft: float[ ]
+ bottomRight: float[ ]

Line

+ pointsArray: ArrayList<float[ ]>
+ pointsNum: int

+ addPoint(mousex: float, mousey: float): void
+ validLine(): boolean
+ isPointOnLineSegment(px: float, py: float , x1: float, y1: float, x2: float, y2: float): boolean