Menu

- title: string
- start: JButton
- tutorial: JButton
- back: JButton
- gamePanel: GamePanel
- TUTORIAL_TEXT: String
- mainMenuPanel: JPanel
- tutorialPanel: JPanel

+ Menu(gamePanel: GamePanel)
- createMainMenuPanel(): JPanel
- createTutorialPanel(): JPanel
- buttonPressed(buttonType: ButtonType): void

GamePanel

+ GAME_WIDTH: int
+ GAME_HEIGHT: int
+ gameThread: Thread
+ image: Image
+ graphics: Graphics
+ playerLeft: Player
+ playerRight: Player
+ bulletLeft: Bullet
+ bulletRight: Bullet
+ score: Score
- obstacle: Obstacle
- powerup: Powerup
+ menu: Menu
− gameStarted: boolean
+ bulletWidth: int
+ bulletHeight: int
+ mapManager: MapManager
- backgroundImage: Buffered Image
- lastBulletClearTIme: long
- BULLET_REST_DELAY: long
- canShoot: boolean
- firstPlayerHasShot: boolean
- secondPlayerHasShot: boolean
- firstShootingPlayer:Player
- secondShootingPlayer: Player
- isLeftPlayerShooting: boolean
- isRightPlayerShooting: boolean
- SHOOT_PAUSE_DURATION: long
- leftPlayerShootStartTime: long
- rightPlayerShootStartTIme:long

+ startGame(): void
+ paint(g: Graphics): void
+ draw(g: Graphics): void
- loadMapAssets(): void
+ move(): void
- handlePlayerShoot(shooter: Player, otherPlayer: Player, isLeftPlayer:boolean): void
- handleBulletCleared(): void
- ClearBullet(): void
- cleanupBullet(): void
+ checkCollision(): void
+ run(): void
- reset(): void
+ keyPressed(e: KeyEvent): void
+ keyReleased(e: KeyEvent): void
+ keyTyped(e: KeyEvent): void

Premade

JPanel
....

JFrame
....

Rectangle
....

EndScreen

- gamePanel: GamePanel
- winnerText: string
- resetButton: JButton
- mainMenuButton: JButton
- back: JButton

+ EndScreen(gamePanel: GamePanel)
- initializeButtons(): void
- createButton(String: text, int: y): JButton
+ showEndScreen(winner: string): void
# paintComponent(g: Graphics): void

PauseMenu

- gamePanel: GamePanel
- pauseButtonImage: Image
- pauseButtonBounds: Rectangle
- resumeButton: JButton
- resetButton: JButton
- mainMenuButton: JButton
- quitButton: JButton
- isPaused: boolean

+ PauseMenu(gamePanel: GamePanel)
- initializeButtons(): void
- createButton(String: text, int: y): JButton
+ togglePause(): void
- setButtonsVisible(visible: boolean): void
paintComponent(g: Graphics): void
+ isPaused(): boolean

GameFrame

+ panel: GamePanel

+ GameFrame()

Main

+ Main(args: String[]): void

<<enumeration>> ButtonType

START
TUTORIAL
BACK

Player

+ SPEED: int
+ SCREEN_HEIGHT: int
- TOPMARGIN: int
- yVelocity: int
- isMoving: boolean
- movementDirection: int
- hasGun: boolean
- playerImageWithGun: BufferedImage
- playerImageNoGun: BufferedImage
- isLeftPlayer: boolean
- SHOOT_PAUSE_DURATION: long
- lastShootTime: long
- isFrozen: boolean
- freezeStartTime: long

+ Player(x: int. y: int, playerWidth: int, screenHeight: int, hasGun: boolean)
- loadPlayerImage(): void
- setYDirection(direction: int): void
+ move(): void
+ shoot(currentTime: long): boolean
+ resumeMovement(currentTime: long): boolean
+ draw(g: Graphics): void
+ setHasGun(hasGun: boolean): void
+ freeze(): void
+ freeze(): void
+ setHasGun(hasGun: boolean): void
+ hasGun: boolean

Bullet

- xVelocity: int
- yVelocity: int
- BASE_SPEED: int
- MIN_HORIZONTAL_RATIO: double
- bulletImage: BufferedImage
- isFromLeftPlayer: boolean
- previousX: int
- previousY: int
- TOP_MARGIN: int
- splitBullets: ArrayList<Bullet>
- scale: double
- hasFreezeEffect: boolean
- playerToUnfreeze: Player

+ Bullet(x: int, y: int, width: int, height:int, isFromLeftPlayer: boolean)
- loadBulletImage(): void
+ move(): void
+ setDirection(double dx, double dy): void
+ bulletBounce(obstacleCenter: Point, obstaclePosition: Point, obstacle: Obstacle, bullet: Bullet): Boolean
+ isOutOfBounds(screenWidth: int): boolean
+ collidesWith(player: Player): boolean
+ createSplitBullets (double angle1, double angle2): void
+ resize(double scaleFactor): void
+ getPreviousX(): int
+ getPreviousY(): int
+ getxVelocity(): int
+ getyVelocity(): int
+ draw(g: Graphics): void

Obstacle

- GAME_WIDTH: int
- GAME_HEIGHT: int
- obstacleImage: BufferedImage
- obstaclePositions: ArrayList<Point>
- brokenObstacles: Map<Point, Long>
- spawnTimes: Map<Point, Long>
- REGENERATION_DELAY: long
- TARGET_OBSTACLES: int
- random: Random
- mapManager: MapManager
- circleRadius: int
- COLLISION_RADIUS_MULTIPLIER: double
- powerup: Powerup
- INVINCIBILITY_DURATION: long

+ Obstacle(gameWidth: int, gameHeight: int, mapManager: MapManager)
- loadObstacleImage(): void
- generateObstaclePositions: void
- generateObstacles(count: int, powerupPositions: ArrayList<Point>): void
+ getCircleCenter(obstaclePosition: Point): Point
+ isPointInCircle(center: Point, test: Point): boolean
+ lineIntersectsCircle(center: Point, lineStart: point, lineEnd: point): boolean
- checkOverlap(newPoint: Point, powerupPositions: ArrayList<Point>): boolean
+ setPowerup(powerup: Powerup): void
+ draw(g: Graphics): void
+ update(powerUpPositions: ArrayList<Point>): void
+ breakObstacle(position: Point): void
- isInvincible(position: Point): boolean
+ getObstaclePositions(): ArrayList<Point>
+ getObstacleImage(): BufferedImage
+ regenerateObstacles(): void

Score

- leftPlayerScore: int
- rightPlayerScore: int

+ Score()
+ scoreLeftPlayer(): void
+ scoreLeftPlayer(): void
+ getLeftPlayerScore(): int
+ getRightPlayerScore(): int
+ draw(g: Graphics, screenWidth: int, screenHeight: int): void
+ reset(): void

MapManager

- currentMap: MapData
- random: Random
+ MAPS: MapData[]

+ MapManager():
+ randomizeMap(): void
+ getBackgroundImage(): String
+ getPowerupImage(): String
+ getObstacleImage(): String

Powerup

- GAME_WIDTH: int
- GAME_HEIGHT: int
- powerupImages: Map<Point, BufferedImage>
- powerupTypes: Map<Point, String>
- powerupPositions: ArrayList<Point>
- usedPowerups: Map<Point, Long>
- spawnTimes: Map<Point, Long>
- REGENERATION_DELAY: long
- TARGET_POWERUPS: string
- random: Random
- mapManager: MapManager
- circleRadius: int
- currentPowerupType: String
- COLLISION_RADIUS_MULTIPLIER: double
- obstacle: Obstacle
- INVINCIBILITY_DURATION: long

+ Powerup(gameWidth: int, gameHeight: int, mapManager, MapManager)
- loadPowerupImage(): void
- generatePowerupPositions (obstaclePositions: ArrayList<Point>): void
- generatePowerups(count: int, obstaclePositions: ArrayList<Point>): void
+ getCircleCenter(powerupPosition: Point): void
+ lineIntersectsCircle(center: Point, lineStart: Point, lineEnd: Point): boolean
+ collidesWith(player: Player): boolean
- checkOverlap(newPoint: Point, obstaclePositions: ArrayList<Point>): boolean
+ setObstacle(obstacle: Obstacle): void
+ draw(g: Graphics): void
+ update(obstaclePositions: ArrayList<Point>): void
+ activatePowerup(position: Point, bullet: Bullet, otherPlayer, Player): String
+ regeneratePowerup(obstaclePositions: ArrayList<Point>): void
- isInvincible(position: Point): boolean
+ getPowerupImage(): BufferedImage
+ getPowerupPositions(): ArrayList<Point>

SoundManager

- backgroundMusic: Clip
- gameStartSound: Clip
- pausePlaySound: Clip
- bulletSound: Clip
- obstacleBounceSound: Clip
- wallBounceSound: Clip
- bombSound: Clip
- freezeSound: Clip
- bigBulletSound: Clip

+ SoundManager(): void
- loadSounds(): void
- loadClip(soundFile: String, volume: float): Clip
- setVolume(clipd: Clip, volume: float): void
+ adjustBackgroundMusicVolume(isMainMenu: Boolean): void
+ playBackgroundMusic(): void
+ stopBackgroundMusic(): void
+ playGameStart(): void
+ playPauseAndPlay(): void
+ playBulletSound(): void
+ playObstacleBounce(): void
+ playBombSound(): void
+ playFreezeSound(): void
+ playBigBulletSound(): void
- playSound(clip:Clip): void
+ cleanup(): void