KillDrLucky Game Design

Model

<<interface>> ReadOnlyWorld

+ getWorldName(): String
+ getRows(): int
+ getCols(): int
+ getSpaces(): List<Room>
+ getSpace(idx: int): Room
+ getTarget(): Target
+ neighborsOf(idx: int): List<Integer>
+ visibleFrom(idx: int): Set<Integer>
+ describeSpace(idx: int): String

«enum» AttackStatus

SUCCESS
NOT_SAME_SPACE
SEEN_BY_OTHERS
NO_SUCH_ITEM
TARGET_ALREADY_DEAD

<<interface>> WorldModel

+ moveTargetNext(): void
+ renderBufferedImage(cellSize: int): BufferedImage
+ canAttack(playerId: int, itemId: int): AttackStatus
+ attack(playerId: int, itemId: int): AttackStatus

<<interface>> GameModelAPI

+ addPlayer(name: String, start: String, isAI: boolean): void
+ movePlayer(name: String, dir: String): String
+ pickUpItem(name: String, item: String): String
+ lookAround(name: String): String
+ describePlayer(name: String): String
+ describeSpace(space: String): String
+ moveTarget(): void
+ autoAction(name: String): String
+ getPlayers(): List<iPlayer>
+ isGameOver(): boolean
+ endGame(): void

AxisAlignedVisibility

+ AxisAlignedVisibility()
+ visibleFrom(spaceIdx: int, world: ReadOnlyWorld): Set

<<interface>> VisibilityStrategy

+ visibleFrom(spaceIdx: int, world: ReadOnlyWorld): Set

World

- name: String
- rows: int
- cols: int
- spaces: List<Room>
- items: List<Item>
- target: Target
- neighbors: Map<Integer, List<Integer>>
- visibilityStrategy: VisibilityStrategy

+ World(data: WorldParser.WorldData, visibilityStrategy: VisibilityStrategy)
+ getWorldName(): String
+ getRows(): int
+ getCols(): int
+ getSpaces(): List<Space>
+ getSpace(idx: int): Space
+ getTarget(): Target
+ neighborsOf(idx: int): List<Integer>
+ visibleFrom(idx: int): Set<Integer>
+ describeSpace(idx: int): String
+ moveTargetNext(): void
+ renderBufferedImage(cellSize: int): BufferedImage
+ canAttack(playerId: int, itemId: int): AttackStatus
+ attack(playerId: int, itemId: int): AttackStatus
+ addPlayer(name: String, start: String, isAI: boolean): void
+ getPlayers(): List<iPlayer>
+ movePlayer(name: String, dir: String): String
+ pickUpItem(name: String, item: String): String
+ lookAround(name: String): String
+ describePlayer(name: String): String
+ describeSpace(space: String): String
+ moveTarget(): void
+ autoAction(name: String): String
+ getPlayers(): List<iPlayer>
+ isGameOver(): boolean
+ endGame(): void
- isSeenByOthers(attacker: int): boolean
- computeNeighbors(): Map<Integer, List<Integer>>

Target

- name: String
- health: int
- currentSpaceIndex: int

+ Target(name: String, health: int, startIndex: int)
+ getName(): String
+ getHealth(): int
+ setHealth(h: int): void
+ takeDamage(amount: int): void
+ isAlive(): boolean
+ getCurrentSpaceIndex(): int
+ setCurrentSpaceIndex(idx: int): void

<<interface>> Character

+ getName(): String
+ getCurrentSpaceIndex(): int
+ setCurrentSpaceIndex(idx: int): void

WorldParser

+ parse(in: InputStream): World
+ parse(path: Path): World

<<interface>> Space

+ getIndex(): int
+ getName(): String
+ getArea(): Rect
+ getItems(): List<Item>

<<interface>> iPlayer

+ isComputerControlled(): boolean
+ getItems(): List<Item>
+ move(direction: String, model: GameModelApi): String
+ pickUp(item: String, model: GameModelApi): String
+ lookAround(model: GameModelApi): String

ComputerPlayer

- random: Random

+ autoAction(model): String

Room

- index: int
- name: String
- area: Rect
- items: List<Item>

+ Room(index: int, name: String, area: Rect, items: List<Item>)
+ getIndex(): int
+ getName(): String
+ getArea(): Rect
+ getItems(): List<Item>

<<interface>> Item

+ getName(): String
+ getDamage(): int
+ getRoomIndex(): int

Player

- name: String
- currentSpaceIndex: int
- items: List<Item>

+ Player(name: String, startIndex: int, List<Item>)
+ getName(): String
+ getCurrentSpaceIndex(): int
+ setCurrentSpaceIndex(idx: int): void
+ getItems(): List<Item>
+ addItem(item: Item): void
+ removeItem(item: Item): void
+ isComputerControlled(): boolean
+ move(direction: String, model: GameModelApi): String
+ pickUp(item: String, model: GameModelApi): String
+ lookAround(model: GameModelApi): String

Point

- row: int
- col: int

+ Point(row: int, col: int)
+ getRow(): int
+ getCol(): int

Rect

- ul: Point
- lr: Point

+ Rect(ul: Point, lr: Point)
+ getUpperleft(): Point
+ getLowerRight(): Point
+ width(): int
+ height(): int
+ contains(p: Point): boolean
+ intersects(r: Rect): boolean

Weapon

- name: String
- damage: int
- roomIndex: int

+ Weapon(name: String, damage: int, roomIndex: int)
+ getName(): String
+ getDamage(): int
+ getRoomIndex(): int

View

<<interface>> View

+ renderWorld(model: GameModelAPI, file: String): void
+ refresh(model: GameModelAPI, out: Appendable): void

GameView

+ renderWorld(model: GameModelAPI, file: String): void
+ refresh(model: GameModelAPI, out: Appendable): void

Controller

GameController

- model: GameModelAPI
- view: View
- in: Readable
- out: Appendable
- maxTurns: int

+ KillDrLuckyController(model: GameModelAPI, view: View, in: Readable, out: Appendable, maxTurns: int)
+ playGame(): void
- processTurn(currentPlayer: iPlayer) void
- getNextPlayer(currentIndex: int): int
- executeCommand(cmd: String, currentPlayer: iPlayer, scanner: Scanner)

<<interface>> Controller

+ playGame(): void