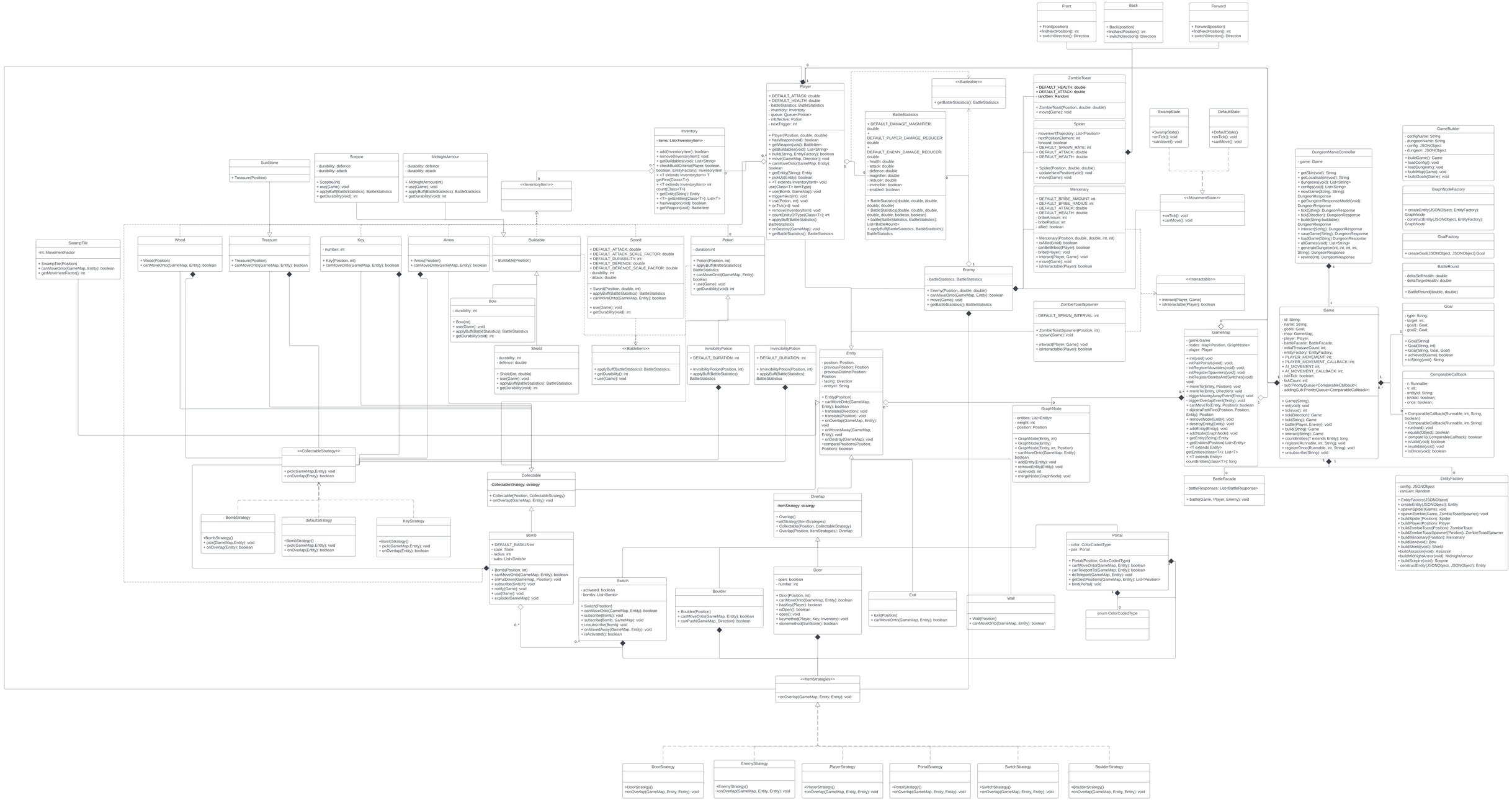
Front
+ Front(position)
+findNextPosition(): int
+ switchDirection(): Direction
Back
+ Back(position)
+findNextPosition(): int
+ switchDirection(): Direction
Forward
+ Forward(position)
+findNextPosition(): int
+ switchDirection(): Direction
Player
+ DEFAULT_ATTACK: double
+ DEFAULT_HEALTH: double
- battleStatistics: BattleStatistics
- inventory: Inventory
- queue: Queue<Proton>
- inEffective: Potion
- nextTrigger: int
+ Player(Position, double, double)
+ hasWeapon(void): boolean
+ getWeapon(void): BattleItem
+ getBuildables(void): List<String>
+ build(String, EntityFactory): boolean
+ move(GameMap, Direction): void
+ canMoveOnto(GameMap, Entity): boolean
+ getEntity(String): Entity
+ pickUp(Entity): boolean
+ <T extends InventoryItem> void use(Class<T> itemType)
+ use(Bomb, GameMap): void
+ triggerNext(int): void
+ use(Potion, int): void
+ onTick(int): void
+ remove(InventoryItem): void
+ countEntityOfType(Class<T>): int
+ applyBuff(BattleStatistics): BattleStatistics
+ onDestroy(GameMap): void
+ getBattleStatistics(): BattleStatistics
<<Battleable>>
+ getBattleStatistics(): BattleStatistics
ZombieToast
+ DEFAULT_HEALTH: double
+ DEFAULT_ATTACK: double
-randGen: Random
+ ZombieToast(Position, double, double)
+ move(Game): void
BattleStatistics
+ DEFAULT_DAMAGE_MAGNIFIER: double
+ DEFAULT_PLAYER_DAMAGE_REDUCER: double
+ DEFAULT_ENEMY_DAMAGE_REDUCER: double
- health: double
- attack: double
- defence: double
- magnifier: double
- reducer: double
- invincible: boolean
- enabled: boolean
+ BattleStatistics(double, double, double, double, double)
+ BattleStatistics(double, double, double, double, double, double, boolean, boolean)
+ battle(BattleStatistics, BattleStatistics): List<BattleRound>
+ applyBuff(BattleStatistics, BattleStatistics): BattleStatistics
+ getBattleStatistics(): BattleStatistics
Spider
- movementTrajectory: List<Position>
- nextPositionElement: int
- forward: boolean
+ DEFAULT_SPAWN_RATE: int
+ DEFAULT_ATTACK: double
+ DEFAULT_HEALTH: double
+ Spider(Position, double, double, double)
- updateNextPosition(void): void
+ move(Game): void
SwampState
+SwampState()
+onTick(): void
+canMove(): void
DefaultState
+DefaultState()
+onTick(): void
+canMove(): void
<<MovementState>>
+onTick(): void
+canMove(): void
Mercenary
+ DEFAULT_BRIBE_AMOUNT: int
+ DEFAULT_BRIBE_RADIUS: int
+ DEFAULT_ATTACK: double
+ DEFAULT_HEALTH: double
- bribeAmount: int
- bribeRadius: int
- allied: boolean
+ Mercenary(Position, double, double, int, int)
+ isAllied(void): boolean
- canBeBribed(Player): boolean
+ bribe(Player): void
+ interact(Player, Game): void
+ move(Game): void
+ isInteractable(Player): boolean
Inventory
- items: List<InventoryItem>
+ add(InventoryItem): boolean
+ remove(InventoryItem): void
+ getBuildables(void): List<String>
+ checkBuildCriteria(Player, boolean, boolean, EntityFactory): InventoryItem
+ <T extends InventoryItem> T getFirst(Class<T>)
+ <T extends InventoryItem> int count(Class<T>)
+ getEntity(String): Entity
+ <T> getEntities(Class<T>): List<T>
+ hasWeapon(void): boolean
+ getWeapon(void): BattleItem
<<InventoryItem>>
SunStone
+ Treasure(Position)
Sceptre
- durability: defence
- durability: attack
+ Sceptre(int)
+ use(Game): void
+ applyBuff(BattleStatistics): BattleStatistics
+ getDurability(void): int
MidnightArmour
- durability: defence
- durability: attack
+ MidnightArmour(int)
+ use(Game): void
+ applyBuff(BattleStatistics): BattleStatistics
+ getDurability(void): int
SwampTile
-int: MovementFactor
+ SwampTile(Position)
+ canMoveOnto(GameMap, Entity): boolean
+ getMovementFactor(): int
Wood
+ Wood(Position)
+ canMoveOnto(GameMap, Entity): boolean
Treasure
+ Treasure(Position)
+ canMoveOnto(GameMap, Entity): boolean
Key
- number: int
+ Key(Position, int)
+ canMoveOnto(GameMap, Entity): boolean
Arrow
+ Arrow(Position)
+ canMoveOnto(GameMap, Entity): boolean
Buildable
+ Buildable(Position)
Sword
+ DEFAULT_ATTACK: double
+ DEFAULT_ATTACK_SCALE_FACTOR: double
+ DEFAULT_DURABILITY: int
+ DEFAULT_DEFENCE: double
+ DEFAULT_DEFENCE_SCALE_FACTOR: double
- durability: int
- attack: double
+ Sword(Position, double, int)
+ applyBuff(BattleStatistics): BattleStatistics
+ canMoveOnto(GameMap, Entity): boolean
+ use(Game): void
+ getDurability(void): int
Potion
- duration:int
+ Potion(Position, int)
+ applyBuff(BattleStatistics): BattleStatistics
+ canMoveOnto(GameMap, Entity): boolean
+ use(Game): void
+ getDurability(void): int
Bow
- durability: int
+ Bow(int)
+ use(Game): void
+ applyBuff(BattleStatistics): BattleStatistics
+ getDurability(void): int
Shield
- durability: int
- defence: double
+ Shield(int, double)
+ use(Game): void
+ applyBuff(BattleStatistics): BattleStatistics
+ getDurability(void): int
<<BattleItem>>
+ applyBuff(BattleStatistics): BattleStatistics;
+ getDurability(); int
+ use(Game): void
InvisibilityPotion
+ DEFAULT_DURATION: int
+ InvisibilityPotion(Position, int)
+ applyBuff(BattleStatistics): BattleStatistics
InvincibilityPotion
+ DEFAULT_DURATION: int
+ InvincibilityPotion(Position, int)
+ applyBuff(BattleStatistics): BattleStatistics
Enemy
- battleStatistics: BattleStatistics
+ Enemy(Position, double, double)
+ canMoveOnto(GameMap, Entity): boolean
+ move(Game): void
+ getBattleStatistics(): BattleStatistics
ZombieToastSpawner
- DEFAULT_SPAWN_INTERVAL: int
+ ZombieToastSpawner(Position, int)
+ spawn(Game): void
+ interact(Player, Game): void
+ isInteractable(Player): boolean
<<Interactable>>
+ interact(Player, Game)
+ isInteractable(Player): boolean
DungeonManiaController
- game: Game
+ getSkin(void): String
+ getLocalisation(void): String
+ dungeons(void): List<String>
+ configs(void): List<String>
+ newGame(String, String): DungeonResponse
+ getDungeonResponseModel(void): DungeonResponse
+ tick(String): DungeonResponse
+ tick(Direction): DungeonResponse
+ build(String buildable): DungeonResponse
+ interact(String): DungeonResponse
+ saveGame(String): DungeonResponse
+ loadGame(String) DungeonResponse
+ allGames(void): List<String>
+ generateDungeon(int, int, int, int, String): DungeonResponse
+ rewind(int): DungeonResponse
GameBuilder
- configName: String
- dungeonName: String
- config: JSONObject
- dungeon: JSONObject
+ buildGame(): Game
+ loadConfig(): void
+ loadDungeon(): void
+ buildMap(Game): void
+ buildGoals(Game): void
GraphNodeFactory
+ createEntity(JSONObject, EntityFactory): GraphNode
- constructEntity(JSONObject, EntityFactory): GraphNode
GoalFactory
+ createGoal(JSONObject, JSONObject):Goal
BattleRound
- deltaSelfHealth: double
- deltaTargetHealth: double
+ BattleRound(double, double)
Goal
- type: String;
- target: int;
- goal1: Goal;
- goal2: Goal;
+ Goal(String)
+ Goal(String, int)
+ Goal(String, Goal, Goal)
+ achieved(Game): boolean
+ toString(void): String
ComparableCallback
- r: Runnable;
- v: int;
- entityId: String;
- isValid: boolean;
- once: boolean;
+ ComparableCallback(Runnable, int, String, boolean)
+ ComparableCallback(Runnable, int, String)
+ run(void): void
+ equals(Object): boolean
+ compareTo(ComparableCallback): boolean
+ isValid(void): boolean
+ invalidate(void): void
+ isOnce(void): boolean
GameMap
- game:Game
- nodes: Map<Position, GraphNode>
- player: Player
+ init(void):void
- initPairPortals(void): void;
- initRegisterMovables(void): void;
- initRegisterSpawners(void): void;
- initRegisterBombsAndSwitches(void): void;
+ moveTo(Entity, Position): void
+ moveTo(Entity, Direction): void
- triggerMovingAwayEvent(Entity): void
- triggerOverlapEvent(Entity): void
+ canMoveTo(Entity, Position): boolean
+ dijkstraPathFind(Position, Position, Entity): Position
+ removeNode(Entity): void
+ destroyEntity(Entity): void
+ addEntity(Entity): void
+ addNode(GraphNode): void
+ getEntity(String):Entity
+ getEntities(Position):List<Entity>
+ <T extends Entity> getEntities(class<T>): List<T>
+ <T extends Entity> countEntities(class<T>): long
Game
- id: String;
- name: String;
- goals: Goal;
- map: GameMap;
- player: Player;
- battleFacade: BattleFacade;
- initialTreasureCount: int;
- entityFactory: EntityFactory;
+ PLAYER_MOVEMENT: int;
+ PLAYER_MOVEMENT_CALLBACK: int;
+ AI_MOVEMENT: int;
+ AI_MOVEMENT_CALLBACK: int;
- isInTick: boolean;
- tickCount: int;
- sub:PriorityQueue<ComparableCallback>;
- addingSub:PriorityQueue<ComparableCallback>;
+ Game(String)
+ init(void): void
+ tick(void): int
+ tick(Direction): Game
+ tick(String): Game
+ battle(Player, Enemy): void
+ build(String): Game
+ interact(String): Game
+ countEntities(T extends Entity): long
+ register(Runnable, int, String): void
+ registerOnce(Runnable, int, String): void
+ unsubscribe(String): void
BattleFacade
- battleResponses: List<BattleResponse>
+ battle(Game, Player, Enemy): void
EntityFactory
- config: JSONObject
- ranGen: Random
+ EntityFactory(JSONObject)
+ createEntity(JSONObject): Entity
+ spawnSpider(Game): void
+ spawnZombie(Game, ZombieToastSpawner): void
+ buildSpider(Position): Spider
+ buildPlayer(Position): Player
+ buildZombieToast(Position): ZombieToast
+ buildZombieToastSpawner(Position): ZombieToastSpawner
+ buildMercenary(Position): Mercenary
+ buildBow(void): Bow
+ buildShield(void): Shield
+buildAssassin(void): Assassin
+buildMidnightArmour(void): MidnightArmour
+ buildSceptre(void): Sceptre
- constructEntity(JSONObject, JSONObject): Entity
<<CollectableStrategy>>
+ pick(GameMap,Entity): void
+ onOverlap(Entity): boolean
Collectable
-CollectableStrategy: strategy
+ Collectable(Position, CollectableStrategy)
+ onOverlap(GameMap, Entity): void
BombStrategy
+BombStrategy()
+ pick(GameMap,Entity): void
+ onOverlap(Entity): boolean
defaultStrategy
+BombStrategy()
+ pick(GameMap,Entity): void
+ onOverlap(Entity): boolean
KeyStrategy
+BombStrategy()
+ pick(GameMap,Entity): void
+ onOverlap(Entity): boolean
Bomb
+ DEFAULT_RADIUS:int
- state: State
- radius: int
- subs: List<Switch>
+ Bomb(Position, int)
+ canMoveOnto(GameMap, Entity): boolean
+ onPutDown(Gamemap, Position): void
+ subscribe(Switch): void
+ notify(Game): void
+ use(Game): void
+ explode(GameMap): void
Entity
- position: Position
- previousPosition: Position
- previousDistinctPosition: Position
- facing: Direction
- entityId: String
+ Entity(Position)
+ canMoveOnto(GameMap, Entity): boolean
+ translate(Direction): void
+ translate(Position): void
+ onOverlap(GameMap, Entity): void
+ onMovedAway(GameMap, Entity): void
+ onDestroy(GameMap): void
+comparePositions(Position, Position): boolean
GraphNode
- entities: List<Entity>
- weight: int
- position: Position
+ GraphNode(Entity, int)
+ GraphNode(Entity)
+ GraphNode(Entity, int, Position)
+ canMoveOnto(GameMap, Entity): boolean
+ addEntity(Entity): void
+ removeEntity(Entity): void
+ size(void): int
+ mergeNode(GraphNode): void
Overlap
-itemStrategy: strategy
+ Overlap()
+setStrategy(ItemStrategies)
+ Collectable(Position, CollectableStrategy)
+ Overlap(Position, ItemStrategies): Overlap
Portal
- color: ColorCodedType
- pair: Portal
+ Portal(Position, ColorCodedType)
+ canMoveOnto(GameMap, Entity): boolean
+ canTeleportTo(GameMap, Entity): boolean
+ doTeleport(GameMap, Entity): void
+ getDestPositions(GameMap, Entity): List<Position>
+ bind(Portal): void
enum ColorCodedType
Switch
- activated: boolean
- bombs: List<Bomb>
+ Switch(Position)
+ canMoveOnto(GameMap, Entity): boolean
+ subscribe(Bomb): void
+ subscribe(Bomb, GameMap): void
+ unsubscribe(Bomb): void
+ onMovedAway(GameMap, Entity): void
+ isActivated(): boolean
Boulder
+ Boulder(Position)
+ canMoveOnto(GameMap, Entity): boolean
+ canPush(GameMap, Direction): boolean
Door
- open: boolean
- number: int
+ Door(Position, int)
+ canMoveOnto(GameMap, Entity): boolean
+ hasKey(Player): boolean
+ isOpen(): boolean
+ open(): void
+ keymethod(Player, Key, Inventory): void
+ stonemethod(SunStone): boolean
Exit
+ Exit(Position)
+ canMoveOnto(GameMap, Entity): boolean
Wall
+ Wall(Position)
+ canMoveOnto(GameMap, Entity): boolean
<<ItemStrategies>>
+onOverlap(GameMap, Entity, Entity): void
DoorStrategy
+DoorStrategy()
+onOverlap(GameMap, Entity, Entity): void
EnemyStrategy
+EnemyStrategy()
+onOverlap(GameMap, Entity, Entity): void
PlayerStrategy
+PlayerStrategy()
+onOverlap(GameMap, Entity, Entity): void
PortalStrategy
+PortalStrategy()
+onOverlap(GameMap, Entity, Entity): void
SwitchStrategy
+SwitchStrategy()
+onOverlap(GameMap, Entity, Entity): void
BoulderStrategy
+BoulderStrategy()
+onOverlap(GameMap, Entity, Entity): void