Initial UML Diagram

Group 4

These are the classes that the team decided upon once we understood more about what was required by our game application. However as we began working on implementing these, we realised that some additional functions and classes were required to aid implementation.

Entity
- loc: Point

+ move(Move, Map): abstract boolean
+ getLoc: Point

Move
- dir: char
- undo: boolean
- entityMoved: Entity

+ isUndo: boolean
+ getEntityMoved: Entity
+ setEntityMoved(Entitiy): void
+ setUndo: void
+ getNewPoint(Point): Point

MainUI
-ANIMATION, BUTTONS, RESOURCES and GRAPHICS AS NEEDED

GameUI
-ANIMATION, BUTTONS, RESOURCES and GRAPHICS AS NEEDED

TutorialUI
-ANIMATION, BUTTONS, RESOURCES and GRAPHICS AS NEEDED

MainApplication
+ Height: int
+ Width: int

+ main(String) : void
+ start(Stage) : void

Box
+ move(Move, Map): boolean

Player
+ move(Move, Map): boolean

GameState
- map : Block[][]
-goals : List<Point>
- boxes : List<Box>
- player : Point
- ID : int
- cost : int
- stateCode : double

+ calculateID() : void
+ calculateStateCode() : void
+ expandState() : void
+ findBox() : Box
+ findStartingPoints() : ArrayList<Block>
+ getCost : int
+ getID : int
+ getPossibleStates() : ArrayList<GameStates>

Game
- map: Map
- p: Player
- prevMoves: Deque<Move>
- height : int
- width : int

<<constructor>> + Game(Map)
+ getGrid: int[][]
+ getMap: Map
+ getPlayer: Player
+ move(char): boolean
+ undoMove: boolean
+ getBoxLocs: List<Point>
+ getGoalLocs : List<Point>
+ getHeight : int
+ getWidth : int
+ move(dir : chair) : void
+ undoMove() : boolean
+ updateMap(int, int, int) : void

Block
- type: int
- i: int
- j: int
- canGoTo: ArrayList<Block>

+ getType: int
+ setType (int): void
+ getI: int
+ getJ: int
+ getConnectingBlocks:ArrayList<Block>
+ addCanGoTo(Block):void

Map Generator
- puzzleB: Block[][]
- ts: ArrayList<Template>
- goals: ArrayList<Block>
- boxes: ArrayList<Block.
- playerLocation: Point
- boxLocs: List<Box>
-goalLocs: List<Point>

+ addPlayer: void
+ addGoal: void

Map
- player: Player
- map: int[][]
boxLocs: List<Box>
goallocs: List<Point>

<<constructor>> + Map(Map Generator)
+ getMap: int[][]
+ getBoxLocs: List<Box>
+ getGoallocs: List<Point>
+ getPlayer; Player
+ updateMap(int, int, int): void
+ getBox(Point): Box
+ isFreeSpace(Point): boolean
+ isBox(Point): boolean
+ isGoal(Point): boolean

Template
- block: Block[][]
- template : int[][]

<<constructor>> + Template(int)
+ template_x : void

AStar
- boxes: List<Box>
- player: Point
- end: Block
- map: Block[][]
- queue: PriorityQueue<State>

+ search(): boolean
+ canGoTo(int,int) : void
+ expandState(State) : void
+ createInitialStates() : void
+ currState() : void

State
- f: int
- g: int
- h:int
- end: Block
- path: ArrayList<Block>

+ getF: int
+ containsEndState: boolean
+ expandState: ArrayList<State>
+ getPath: ArrayList<Block>
+ addBlocks(ArrayList<Block>): void
+ calculateF: void
+ calculateG: int
+ calculateH: int
+ addBlock(Block): void