Geometry Utility Library

Unit Vector

-xHat : final double
-yHat : final double

+ UnitVector(double x, double y) : Constructor
+ UnitVector(double direction): Constructor
+UnitVector(UnitVector direction):Constructor
+xHat() : double
+yHat(): double

Point

-xCoord : final double
-yCoord : final double

+Point(double x, double y) :Constructor
+Point(Point point) : Constructor
+ getX() : double
+ getY() : double
+inside(Square aSqaure) : boolean
+translate(UnitVector direction, double distance) : Point

Square

-Point center : final
-Point sideLength :final double
-Point bottomLeft : final
-Point bottomRight : final
-Point topLeft: final
Point topRight : final

+ Square(Point theCenter, double theSideLength) : Constructor
+ Square(Square theSquare): Constructor
+ getCenter(): Point
+ getSideLength() : double
+ getBottomLeft() : Point
+getTopLeft() : Point
+ getTopRight(): Point
+ intersects(Square otherSqaure):boolean
+ translate(UnitVector direction, double distance) : Square

Cat Game

-String[] clargs : static
-nextLevel : int
-currentScore : int
-currentGameState : GameState
-primaryStage : Stage

+ main(String[ ] args): void
+launchLevel() :void
-showLoseScreen() : void
-showQuitScreen() : void
-showWinScreen() : void
+returnToApp() : void
+start(Stage primaryStage):void

Text Cat Game

+ main(String[ ] args): void

ImageManager

-catImage: Image
-dogImage: Image
-keyImage: Image
-treasureImage: Image
-wallImage: Image
-floorImage: Image
-closedGateImage: Image
-openGateImage: Image

+loadImages :void
+getCatImage() : Image
+getDogImage() : Image
+getKeyImage(): Image
+getTreasureImage(): Image
+getWallImage() : Image
+getFloorImage(): Image
+getClosedGateImage(): Image
+getOpenGateImage(): Image

GameState

-theHero: Hero
-theGameMap: Map
-theScreen : Screen
-theHeatMap: HeatMap
-theUpdater: Updater
-theCollider: Collider
-theImageManager: ImageManager
-gameSpeed : double
-score : int
-mainApp: CatGame
-isLost: boolean
-isWon: boolean
-isDone: boolean
-haskeys: boolean

+ GameState(aScore : int);
+ getMap() : Map
+ setMap(aMap : Map): void
+ getUpdater() : Updater
+ setUpdater(anUpdater : updater) : void
+ getCollider() : Collider
+ setCollider(aCollider : Collider) : void
+ getImageManager() : ImageManager
+ setImageManager(theManager : ImageManager) :void
+ setHeatMap(aMap : HeatMap): void
+ getHeatMap() : HeatMap
+ getScreen(): Screen
+ setScreen(aScreen : Screen): void
+ getScore() : int
+ addToScore(num : int) : void
+ getHero(): Hero
+ setHero(aHero : Hero): void
+ isDone(): boolean
+ getGameSpeed() : double
+ isWon(): boolean
+ isLost() : boolean
+ hasKeys() : boolean
+ setHasKeys() : void
+ heroDied() : void
+ getMainApp: CatGame
+ heroEscaped(): void
+ win() : void
+ lose() : void
+ quit() : void

LevelLoader

+ load(String filename, int startingScore): GameState
+prepareGamestate(theGameState: GameState , levelScanner :Scanner) : void
+buildMap(theGameState: GameState, levelScanner:Scanner : void

Keyboard Handler

-controller : HeroController
-theGameState : GameState
-keyPressed : boolean
-currentKey : KeyCode
-UPCHAR:KeyCode
-DOWNCHAR:KeyCode
-RIGHTCHAR:KeyCode
-LEFTCHAR:KeyCode
-PAUSECHAR:KeyCode
-QUITCHAR:KeyCode

+ Direction: enum
+ KeyboardHandler(aGameState :GameState) :Constructor
+setController(aController :HeroController) : void
-handleKeyPressed(event:KeyEvent) : void
-handleKeyReleased(event : KeyEvent) :void
+ handle(even : KeyEvent): void

Text Keyboard Handler

-controller : HeroController
-theGameState : GameState
-keyPressed : boolean
-currentKey : KeyCode
-UPCHAR:KeyCode
-DOWNCHAR:KeyCode
-RIGHTCHAR:KeyCode
-LEFTCHAR:KeyCode
-PAUSECHAR:KeyCode
-QUITCHAR:KeyCode

+ Direction: enum
+ Text KeyboardHandler(aGameState :GameState, aController:HeroController, aScanner : Scanner) :Constructor
+handleInput() : void

Map

-length : int
-height : int
-theMap : Tile[][]

+ Map (int length, int height) : Constructor
+ getLength() : int
+ getHeight() : int
+ setTile(int x, int y, Tile atile): void
+ Tile getTile(int x, int y) :Tile

TextUpdater

+handle(deltaT : double) : void
+TextUpdater (aGameState:GameState) : Constructor
+getPriority(object :Updatable) : int
+registerUpdatable(object : Updatable) : void
-cullDead() : void
-checkForGameEnd() : void

Updater

+theUpdateQueue : theUpdateQueue
+timer : GameTimer
+theGameState : GameState
+isPaused : boolean

-GameTimer extends AnimationTimer : class
+handle(now:long) : void
-UpdateQueue extends TreeSet<Updatable> : class
+Updater(aGameState:GameState) : Constructor
-getPriority(object Updatable): int
+registerUpdatable : void
+pause():void
+play():void
+togglePause():void
-checkForGameEnd():void

Updatable

theGameState : GameState
- theUpdate : Updater
- dead : boolean

+ Updatable(aGameState : GameState): Constructor
+ isDead():: void
+ kill(): void
 # update(deltaT : double): void

Screen

-height: int
-length: int
-ArrayList<Drawable> objectsToDraw
-theGameState : GameState
-screenGroup : Group
-PIXELCONVERT : static final int
-pauseLabel : Label
-scoreLabel : Label

+ Screen(aGameState : GameState, height : int, length : int): Constructor
+getLength() : int
+getHeight() : int
+getScreenGroup(): Group
+registerDrawable(aDrawable : Drawable) : void
+cullDead() : void
+showLevelTitle : void
+showPause : void
+hidePause : void

TextScreen

-height: int
-length: int
-screenImage : char[][]
-drawableList :ArrayList<Drawable>
-theGameState : GameState
-zBuff : int[][]

+ TextScreen(aGameState : GameState, length : int, heigth : int) :Constructor
+drawGlyph(glyph : char, x : int, y : int, zDepth : int) : void
+drawScreen() : void
+clearScreen() : void
+update(deltaT : double) : void
+registerDrawable(aDrawable : Drawable) : void
+ updateScore() : void
-cullDead() : void
+displayWinMessage(score:int) : void
+displayLooseMessage(score: int) : void
+displayQuitMessage(score : int) : void

HeatMap

-[][] map : double
-tileMap : Map
-length : int
-height : int
-RELAXATION_CONSTANT : double

+ HeatMap(Map amap): Constructor
+ setHeat(): void
+ getHeat(int x, int y): double
+ modifyHeat(double factor, int x, int y): void
+ update(double deltaT): void

Collider

- objectToCollide : ArrayList<Collable>

+ Updatable(aGameState : GameState): Constructor
+ registerCollidable(object : Collidable): void
+ cullDead(): void
 # update(deltaT : double): void

Drawable

 # position : Point
-theScreen: Screen
#imageChanged : boolean
#positionChanged: boolean
-imageView: ImageView
-tileImage : Image
-imageManger: ImageManager

+ Drawable(aGameState : GameState , aPosition : Point) : Contstructor
+ getImageView(): ImageView
+setImageView(): void
+getScreenImage(): Image
+getImageChanged():boolean
+setImageChanged(boolean value) :void
+getPositionChanged() :boolean
+setPositionChanged(boolean value) : void
+getPosition() :Point
+getzDepth() : int

Collidable

 # theGameMap : Map
#BoundsBox : Square
- Collider : Collider

+ Collectables(aGameState : GameState, aPosition : Point, boundsBoxLength : double): Constructor
+ getBoundsBox(): Square
+ collidesWith(aPawn : Pawn): boolean
#collisionWithHero : void

Tile

-defaultHeat : double

+ Tile(int x, int y, double defaultHeat, GameState gameState): Constructor
+ getDefaultHeat(): double
+ setDefaultHeat(double defaultHeat) : void
 # isWall(): boolean
 # isFloor(): boolean
 # heroentered(): void

Collectables

- boundsBoxLength : double

+ Collectables(aGamestate : gameState, position : Point
 # collisionWithHero(): void

Walls

+ Walls(int x, int y , double defaultHeat, GameState gamestate) : Constructor
+ isWall(): boolean
+ isFloor(): boolean
+ getScreenImage(): Image
+ update(double deltaT): void
+ heroentered(): void
+ getzDepth(): int

Floor

+ Floor(GameState gameState, int x, int y, double defaultHeat) :Constructor
+ getScreenImage() : Image
+ isWall() : boolean
+ isFloor : boolean
+ update(double deltaT) : void
+ herEntered() : void
+ getzDepth() : int

Gates

-isopen : boolean

+ Gates(GameState gameState, int x, int y, double defaultHeat): Constructor
+ isWall(): boolean
+ isFloor(): boolean
+ getScreenImage(): Image
+ update(double deltaT): void
+ heroEntered(): void
+ getzDepth(): int

Treasure

+ Treasure(aGameState : GameState, aPosition : Point): Constructor
+ collisionWithHero(): void
+ update(deltaT : double) : void
+ getScreenGlyph(): char
+ getzDepth(): int

Key

-aKey : boolean
-drawKey : char

+ Key(aGameState : GameState, aPosition : Point): Constructor
+ update(deltaT : double): void
+ collisionWithHeor(): void
+ getScreenImage(): Image
+ getzDepth(): int

Pawn

controller : PawnController

+ Pawn(aGameState : GameState, position : Point, boundsBoxLength : double, speed : double): Constructor
+ getController(): PawnController
+ getSpeed(): double
+ setSpeed(speed : double): boid
+ setcontroller(aController : Pawncontroller): void
+ canGo(direction : UnitVector, distance : double): bealoon
+ move (direction : UnitVector, distance : double): void
+ update(deltaT : double): void

Pawn Controller

 # nextTile : Tile
 # currentTile : Tile
 # currentX : int
 # currentY : int
 # gameMap : Map
 # gameState : GameState
 # thePawn : Pawn

+ PawnController(GameState aGameState, Map amap, Tile startTile, Pawn aPawn): Constructor
+ getCurrentX(): int
+ getCurrentY(): int
+ getGameMap(): Map
 # update(double deltaT): void
 # getNextPosition(): Tile

Hero

-boundsBoxLength : double
-defaultSpeed : double

+ Hero(aGameState : GameState, position : Point): Constructor
+ kill(): void
+ canEnter(atile : Tile): boolean
+ move(direction : UnitVector, distance : double): void
+ getController(): PawnController
+ collisionWithHero(): void
+ getScreenImage(): Image
+ getzDepth(): int
+getSpeed(): double

Hero Controller

-heatMapmodificationFactor : double
-nextDirection : UnitVector

+ HeroController(aGameState : GameState, aPawn : Pawn): Constructor
+handleInput(inputDirection : keyboardHandler.Direction): void
+getHeatMapModificationFactor(): double
+getNextDirection(distance : double): UnitVector

Text

Dogs

-boundsBoxLength : double
-defaultSpeed : double

+ Dogs(aGameState : GameState, position : Point): Constructor
+ kill(): void
+ canEnter(atile : Tile): boolean
+ collisionWithHero(): void
+ getScreenImage(): char
+ getzDepth(): int

Simple AI Controller

`-heatMapModificationFactor :double`

+ SimpleAIControler (agamestate : GameState, apawn : Pawn): Constructor
+getHeatMapModificationFactor(): double
+getNextDirection(): UnitVector