Wichtigste Methoden der Klassenbibliothek JGameGrid

Module import: from gamegrid import *

Sprites-Bibliothek

Klasse GameGrid (globale Funktionen nach Aufruf von makeGameGrid)

Methode	Aktion
GameGrid(nbHorzCells, nbVertCells, cellSize, color)	erzeugt ein Spielfenster mit der gegebenen Anzahl horizontalen und vertikalen Zellen, Zellengrösse, mit sichtbaren Gitterlinien in gegebener Farbe, mit Navigationsbalken
GameGrid(nbHorzCells, nbVertCells, cellSize, color, bgImagePath)	erzeugt ein Spielfenster mit der gegebenen Anzahl horizontalen und vertikalen Zellen, Zellengrösse, mit Gitterlinien, mit Hintergrundbild, mit Navigationsbalken
GameGrid(nbHorzCells, nbVertCells, cellSize, None, bgImagePath, False)	erzeugt ein Spielfenster mit der gegebenen Anzahl horizontalen und vertikalen Zellen, Zellengrösse, ohne Gitterlinien, mit Hintergrundbild ohne Navigationsbalken
act()	wird nach dem Start des Simulationszyklus periodisch am Ende aller Actor.act() aufgerufen
addActor(actor, location)	fügt den Actor an der gegebenen Position zum Spielfenster hinzu
addKeyListener(listener)	registriert den Tastaturlistener
addMouseTouchListener (onMouseEvent, mask)	registriert die Funktion onMouseEvent(actor, mouse, spot) mit einer OR-Maske: GGMouse.lClick, GGMouse.lDClick, GGMouse.lDrag, GGMouse.lPress, GGMouse.lRelease, (und analog mit r), GGMouse.enter, GGMouse.move, GGMouse.leave
addStatusBar(height)	fügt ein Statusfenster zum Gamegrid
delay(time)	wartet definierte Zeit (in Millisec)
doPause()	unterbricht den Simulationszyklus
doStep()	führt die Simulation Schritt für Schritt durch
doReset()	setzt alle Actor die Startposition und startet die Simulation neu
doRun()	startet den Simulationszyklus
getActors(Actor class)	gibt alle Actors der gegebenen Klasse in einer Liste zurück
getBg()	gibt die Referenz auf GGBackground zurück
getBgColor()	gibt die Hintergrundfarbe zurück
getKeyCode()	gibt den Tastaturcode der letzten gedrückten Taste zurück
getOneActorAt(location)	gibt den ersten Actor in der gegebenen Zelle zurück (Null, falls keiner)
getOneActor(Actor class)	gibt den ersten Actor der gegebenen Klasse zurück (Null, falls keiner)
getRandomEmptyLocation()	gibt eine zufällige leere Zellenlocation zurück
getRandomLocation()	gibt eine zufällige Zellenlocation zurück
hide()	versteckt das Spielfenster ohne es zu schliessen
isAtBorder(location)	gibt *True* zurück, wenn sich die Zelle am Rand des Spielfensters befindet
isEmpty(location)	gibt *True* zurück, wenn die Zelle leer ist
isInGrid(location)	gibt *True* zurück, wenn sich die Zelle innerhalb des Spielfensters befindet
kbhit()	ergibt *True*, wenn eine Taste gedrückt wurde
toLocation(x, y)	gibt die Zelle mit den Pixel-Koordinaten x und y zurück
openSoundPlayer("wav/ping.wav")	stellt eine Sounddatei bereit. Folgende Sounds sind im tigerjython.jar vorhanden: bird.wav, boing.wav, cat.wav, click.wav, explore.wav, frog.wav, notify.wav
play()	spielt den bereitgestellten Sound ab
refresh()	aktualisiert das Spielfenster

registerAct(onAct)	registriert den Callback onAct, der in jedem Simulationszyklus aufgerufen wird
registerNavigation(started=onStart, stepped=onStep, paused=onPause, resetted = onReset, periodChanged = onPerionChange)	registriert die Callbacks onStart, onStep, onPause, onReset, onPeriodChange, die bei angezeigtem Navigationsbalken aufgerufen werden (nicht alle nötig)
removeActor (actor)	entfernt den Actor vom Spielfenster
removeActorsAt(location)	entfernt alle Actors, die sich in der angegebenen Zelle befinden
removeAllActors()	entfernt alle Actors vom Spielfenster
reset()	setzt die definierte Simulation in die Ausgangsposition zurück, mit Ausnahme der Actors, welche entfernt worden sind
show()	zeigt das Spielfenster an
setBgColor(color)	setzt die Hintergrundfarbe
setSimulationPeriod (milisec)	setzt die Periode des Simulationsloops
setStatusText(text)	setzt den Text in die Statusbar
setTitle(text)	setzt den Titel in der Fenster-Titelleiste

Klasse Actor

Actor(spritepath)	erzeugt einen Actor mit dem gegebenen Spritebild
Actor(True, spritepath)	erzeugt einen rotierbaren Actor mit dem gegebenen Spritebild
Actor(spritepath, nbSprites)	erzeugt einen Actor mit mehreren Sprites (index _0, _1,... e.g..fish_0.gif , fish_1.gif,...)
Actor(True, spritepath, nbSprites)	erzeugt einen rotierbaren Actor mit mehreren Spritebildern
act()	wird nach dem Start des Simulationszyklus periodisch aufgerufen
addActorCollisionListener(listener)	registriert den Kollisionslistener
addCollisionActor(actor)	registriert den Kollisionspartner
addMouseTouchListener (listener)	registriert den MouseTouchListener
collide(actor1, actor2)	Callback beim Auftreten einer Kollision, gibt die Anzahl der Simulationszyklen zurück, während den weitere Events unterdrückt werden
getCollisionActors()	gibt eine Liste der Kollisionskandidaten zurück
getDirection()	gibt die Bewegungsrichtung zurück
getIdVisible()	gibt Id des sichtbaren Sprites zurück
getNeighbours(distance)	gibt eine Liste aller Actors zurück, die sich in der gegebenen Distanz befinden. Die Distanz definiert einen Kreis um den Mittelpunkt der aktuellen Zelle (Einheit: Zellenbreite). Es werden alle Actors, deren Zelle geschnitten wird, zurückgegeben
getNextMoveLocation (location)	gibt die *location* nach dem nächsten *move()* zurück
getX()	gibt die aktuelle horizontale Zellenkoordinate zurück
getY()	gibt die aktuelle vertikale Zellenkoordinate zurück
hide()	macht den Actor unsichtbar, entfernt ihn aber nicht. Nach *reset()* wird er wieder sichtbar
isInGrid()	gibt *True* zurück, wenn sich der Actor innerhalb des Spielfensters befindet
isHorzMirror()	gibt *True* zurück, wenn die Figur horizontal gespiegelt ist
isVertMirror()	gibt *True* zurück, wenn die Figur vertikal gespiegelt ist
isMoveValid()	gibt *True* zurück, wenn ein *move()* den Actor innerhalb des Spielfensters belässt
isNearBorder()	gibt *True* zurück, wenn sich der Actor am Rand des Spielfensters befindet

isVisible()	gibt *True* zurück, wenn der Actor sichtbar ist
move()	bewegt den Actor mit der aktuellen Richtung in eine benachbarte Zelle
move(distance)	bewegt den Actor um die gegebene Distanz
reset()	aufgerufen, wenn der Actor zum Gamegrid hinzugefügt od. Reset-Button gedrückt wird
setCollisionCircle (spriteId,center, radius)	legt den Kreis innerhalb des Actors fest, der für Kollisionen verwendet wird
setCollisionLine(spriteId, startPoint, endPoint)	legt eine Linie innerhalb des Actors fest, die für Kollisionen verwendet wird
setCollisionRectangle(spriteId, center, width, height)	legt das Rechteck innerhalb des Actors fest, das für Kollisionen verwendet wird
setCollisionSpot(spriteId, spot)	legt den Punkt innerhalb des Actors fest, der für Kollisionen verwendet wird
setCollisionImage(spriteId)	wählt für die Kollision nicht transparente Pixel . Nur verfügbar, wenn der Partner spot, line oder circle verwendet
setHorzMirror(True)	spiegelt das Bild horizontal
setVertMirror(True)	spiegelt das Bild vertikal
setSlowDown(factor)	verlangsamt den Aufruf der Methode *act()* des Actors mit dem gegebenen Faktor
setLocation(location)	setzt den Actor in die gegeben Zelle
setLocationOffset(point)	verschiebt die Mitte des Spritebild relativ zur Zellenmitte (location nicht verändert)
setPixelLocation(location)	setzt den Actor auf die gegebene Pixelkoordinate (location/offset werden angepasst)
setX(x)	setzt die x-Koordinate auf den angegebenen Wert
setY(y)	setzt die y-Koordinate auf den angegebenen Wert
show()	Sprite mit der id 0 wird sichtbar
show(spriteId)	Sprite mit der angegebener id wird sichtbar
showNextSprite ()	zeigt das nächste Sprite-Bild (spriteId wird um 1 erhöht (modulo nbSprites))
showPreviousSprite()	zeigt das vorangehende Sprite-Bild (spriteId -1 wird nbSprites - 1)
removeSelf()	der Actor wird vernichtet. Nach *reset()* erscheint er nicht mehr
reset()	wird bei GameGrid.addActor() aufgerufen und wenn der Reset-Button geklickt wird
turn(angle)	ändert die Bewegungsrichtung um den gegebenen Winkel (in Grad im Uhrzeigersinn)

Klasse Location

Location(x, y)	erzeugt Location Objekt mit gegebenen horizontalen und vertikalen Zellenkoordinaten
Location(location)	erzeugt Location Objekt mit der gegebenen Location (clone)
clone()	gibt neue Location mit den gleichen Koordinaten zurück
equals(location)	gibt True zurück, falls die aktuelle Location identisch mit der übergegebenen ist
get4CompassDirectionTo(location)	gibt eine Liste mit 4 benachbarten Location (WEST, EAST, NORTH, SOUTH) zurück
getCompassDirectionTo(location)	gibt eine Liste mit 8 benachbarten Locations (auch diagonal)
getDirectionTo(location)	gibt die Richtung von der aktuellen zu der gegebenen Position in Grad zurück (0 Grad = EAST)

getNeighbourLocation(direction)	gibt eine der 8 benachbarten Zellen zurück. Es wird die Zelle genommen, die am nächsten bei der gegebener Richtung liegt
getNeighbourLocations(distance)	gibt Liste mit allen Zellen mit Zentren innerhalb der gegebenen Distanz zurück
getX()	gibt die aktuelle horizontale Zellenkoordinate zurück
getY()	gibt die aktuelle vertikale Zellenkoordinate zurück

Klasse GGBackground

clear()	löscht den Hintergrund und übermal ihn mit aktueller Hintergrundfarbe
clear(color)	löscht den Hintergrund und übermalt ihn mit der gegebenen Farbe
drawCircle(center, radius)	zeichnet einen Kreis mit dem gegebenen Mittelpunkt und Radius (Pixelkoordinaten)
drawLine(x1,y1, x2, y2)	zeichnet eine Strecke mit gegebenen Endpunkten (Pixelkoordinaten)
drawLine(pt1, pt2)	zeichnet eine Strecke mit den gegebenen Endpunkten (Pixelkoordinaten)
drawPoint(pt)	zeichnet einen Punkt (Pixelkoordinaten)
drawRectangle(pt1, pt2)	zeichnet ein Rechteck mit den gegebenen Diagonaleckpunkten (Pixelkoordinaten)
drawText(text, pt)	schreibt Text an die Position mit dem gegeben Anfangspunkt (Pixelkoordinaten)
fillCell(location, color)	füllt die gegebene Zelle mit der gegebenen Farbe
fillCircle(center,radius)	zeichnet gefüllten Kreis mit dem gegebenen Mittelpunkt und Radius (Pixelkoordinaten)
getBgColor()	gibt die aktuelle Hintergrundfarbe zurück
getColor(location)	gibt die Hintergrundfarbe im Zentrum einer Zelle zurück (Actors unberücksichtigt)
save()	speichert den aktuellen Hintergrund. Wiederherstellen mit restore()
setBgColor(color)	ändert die Hintergrundfarbe
setFont(font)	setzt die Schriftart
setLineWidth(width)	setzt die Liniendicke
setPaintColor(color)	setzt die Zeichnungsfarbe
setPaintMode()	es wird ohne Rücksicht auf den vorhandenen Hintergrund gezeichnet
setXORMode(color)	bei zweimaligen Zeichnen wird er den Hintergrund wieder herstellt
restore()	holt den vorher gespeicherten Hintergrund zurück

Vollständige Dokumentation der Java-Klassenbibliothek JGameGrid : JGameGridDoc