

Auf einen Blick

Klassen, Methoden und Objekte

class Basisklasse	legt Klasse Basisklasse an
class Unterklasse(Klasse)	legt Unterklasse mit Basisklasse an
def __init__(self)	legt wichtigste Parameter fest
Klasse.__init__(self)	Unterklasse erbt __init__-Funktion der Basisklasse
self.eigenschaft	Eigenschaft, die sich auf das Objekt selbst bezieht
objekt = Klasse()	legt Objekt einer Klasse an
objekt.methode()	nutzt Methode einer Klasse, für ein Objekt

Spielfenster

from gamegrid import *	Importiert das Modul GameGrid
makeGameGrid(10, 10, 60)	makeGameGrid(10, 10, 60, None)
legt leeres schwarzes Fenster an (10 x 10 Kästchen mit 60 Pixel Seitenlänge) mit Buttons für Spielsteuerung	
makeGameGrid(10, 10, 60, False)	makeGameGrid(10, 10, 60, None, False)
ohne Buttons für Spielsteuerung	
makeGameGrid(10, 10, 60, Color.white)	mit Buttons für Spielsteuerung und weißem Gitter
makeGameGrid(10, 10, 60, Color.white, "hintergrundbild.png")	mit Hintergrundbild und weißem Gitter, mit Buttons für Spielsteuerung

makeGameGrid(10, 10, 60, None, "bild.png", False)
mit Hintergrundbild, ohne Gitter, und ohne Buttons für Spielsteuerung
setTitle("Mein Spielfenster")
definiert Titel des Spielfensters
setBgColor(255, 255, 255)
definiert Hintergrundfarbe des Fensters
show()
zeigt Spielfenster an
addStatusBar(25)
fügt Statusleiste Breite 25 an
setStatusText("text")
setzt Statustext

Figuren (Actors)

class Apfel(Actor)
definiert Figur Apfel in Basisklasse Actor
Actor.__init__(self, "bild.png")
ordnet Bilddatei der Figur zu
Actor.__init__(self, True, "bild.png")
Figur dreht sich zusätzlich in Bewegungsrichtung
elstar = Apfel()
legt Figur der Klasse Apfel an
addActor(elstar, Location(1, 3), 0)
addActor(Apfel(), Location(1, 3), 90)
platziert Figur der Klasse Apfel im Kästchen mit den Koordinaten X = 1 und Y = 3 und richtet sie nach links bzw. unten aus
addActor(Apfel(), getRandomEmptyLocation())
platziert Figur der Klasse Apfel in einem zufällig ausgewählten, leeren Kästchen
hide()
macht Figur unsichtbar
removeSelf()
löscht Figur aus Spielfenster

Bewegte Figuren

def act(self)	steuert Bewegung
self.move()	bewegt Figur ins nächste Kästchen
self.move(3)	bewegt Figur ins dritte Kästchen
self.getX()	aktuelle X-Position ermitteln
self.turn(180)	Drehung um 180 Grad
self.setHorzMirror(True)	spiegelt Figur horizontal und wieder zurück
self.setHorzMirror(False)	
self.setVertMirror(True)	spiegelt Figur vertikal und wieder zurück
self.setVertMirror(False)	
setSimulationPeriod(100)	setzt Simulationszyklus auf 100 Millisekunden
self.setLocation(Location(0, 1))	setzt Figur in Kästchen mit den Koordinaten X = 0, Y = 1
isAtBorder(Location(self.getX(), self.getY()))	liefert True, sobald Figur Randspalte/-zeile erreicht
setDirection(180)	setzt Richtung auf 180 (links)
doRun()	Figur in Bewegung setzen

Tastatursteuerung und Kollision im Gitter

getOneActorAt(self.getLocation(), Kugel)	
prüft ob sich im aktuellen Kästchen eine Figur der Klasse Kugel befindet	
onKeyPressed(e)	registriert Tastendruck (Ereignis)
makeGameGrid(10, 10, 60, keyPressed = onKeyPressed)	
weist Funktion dem Parameter keyPressed zu	
e.getKeyCode()	fragt ab, welche Taste gedrückt wurde

Figuren animieren

<code>Actor.__init__(self, True, "bild.png", 3)</code>	ermöglicht Wechsel zwischen drei Varianten des Bildes (bild_0.png, bild_1.png, bild_2.png)
<code>showNextSprite()</code>	wechselt zur nächsten Variante des Bildes
<code>if isInGrid(Location(13, 2))</code>	liefert True, wenn sich das angegebene Kästchen innerhalb des Spielfensters befindet
<code>if not self.isInGrid()</code>	liefert True, wenn sich die (animierte) Figur selbst innerhalb des Spielfensters befindet
<code>if self.isNearBorder()</code> <code>if figur.isNearBorder()</code>	liefert True, wenn sich eine (animierte) Figur in der Randspalte oder Zeile des Spielfensters befindet

Maus-Events

<code>onMousePressed(e)</code>	Rückruffunktionen für Maus-Events
<code>onMouseDragged(e)</code> <code>onMouseReleased(e)</code>	
<code>toLocationInGrid(e.getX(), e.getY())</code>	definiert eine Position, die einer Variablen zugewiesen werden kann
<code>makeGameGrid(8, 6, 60, None, "bild.png", False, mousePressed = onMousePressed)</code>	weist Funktionen dem Parameter mousePressed zu
<code>delay(800)</code>	Verzögerung von 800 Millisekunden
<code>removeActor(figur)</code>	löscht Figur
<code>while not isDisposed()</code>	startet Endlosschleife, die bis zum Schließen des Fensters läuft

Maus-Events mit MouseTouchListener

<code>def mouseTouched(actor, mouse, spot)</code>	definiert, was bei einem Maus-Event geschehen soll
<code>mouse.getEvent()</code>	fragt Art des Maus-Events ab
<code>e = GGMouse.lPress</code> <code>e = GGMouse.lRelease</code> <code>e = GGMouse.lDrag</code> <code>e = GGMouse.lClick</code> <code>e = GGMouse.lDClick</code>	<code>e = GGMouse.rPress</code> <code>e = GGMouse.rRelease</code> <code>e = GGMouse.rDrag</code> <code>e = GGMouse.rClick</code> <code>e = GGMouse.rDClick</code>
	ordnet Maus-Event (linke/rechte Taste) einer Variablen zu
<code>pos = Point(mouse.getX(), mouse.getY())</code>	ermittelt aktuelle Position der Maus
<code>a.setPixelLocation(pos)</code>	Mitte des Bildes der Figur a wird auf die Spitze des Mauszeigers an der zuvor ermittelten Position pos geschoben
<code>a.setLocationOffset(Point(0, 0))</code>	Bild der Figur a wird in der Mitte der Zelle positioniert, in der sich Mauszeiger gerade befindet
<code>refresh()</code>	bewirkt fortwährendes Rendern des Bildes, solange Maus-Events registriert werden
<code>apfel1.addMouseListener(mouseTouched, GGMouse.lPress GGMouse.lRelease GGMouse.lDrag, True)</code>	verknüpft Klasse MouseTouchListener mit der Figur apfel und ordnet Ereignisse zu
GUI-Elemente im Spielfenster	
<code>class MeinButton(GGButton, GGButtonListener)</code>	definiert Klasse für Button
<code>def __init__(self, imagePath)</code> <code>GGButton.__init__(self, imagePath)</code>	definiert Button, stets mit Parametern self und imagePath
<code>self.addButtonListener(self)</code>	verknüpft Button mit Klasse GGButtonListener

<code>buttonClicked(self, button)</code>	Methoden zur Steuerung mit Standard-Parametern
<code>buttonPressed(self, button)</code> <code>buttonReleased(self, button)</code>	
<code>button = MeinButton("button.png")</code>	weist Bild mit 2 Varianten zu (button_0.png, button_1.png)
<code>addActor(button1, Location(1, 3))</code>	fügt Button an Position X = 1 und Y = 3 ein
<code>class MeinText(TextActor)</code>	erstellt Klasse für Textfeld
<code>def __init__(self, text)</code>	definiert Textfeld, stets mit Parametern self und text
<code>TextActor.__init__(self, text, Color.white, Color.blue, Font("SanSerif", Font.PLAIN, 48))</code>	definiert Aussehen und Größe des Textes im Textfeld
<code>ausgabe = inputString("Text eingeben")</code>	erzeugt Eingabefeld für Text
<code>addActor(MeinText(ausgabe), Location(0, 1))</code>	platziert Textfeld im Spielfenster

Pixelkollision

<code>class Abprallen(GGActorCollisionListener)</code>	erzeugt Klasse für die Folgen der Berührung zweier Figuren
<code>def collide(self, actor1, actor2)</code>	definiert Berührungsfolgen, stets mit diesen Parametern
<code>return 10</code>	Anzahl Simulationszyklen bis zur nächsten Kollision
<code>abprallen = Abprallen()</code>	
<code>figur_1.addActorCollisionListener(abprallen)</code>	Klasse mit einer Figur verknüpfen
<code>figur_1.addCollisionActor(figur_2)</code>	Figurenpaare für Registrieren der Berührung zuordnen