GameGrid - Hinweise zu Tastatur-Events und GameGrid- Zellen

Die Sache mit den (Tastatur-)Events

Diese Ausführungen beziehen sich auf das Beispiel Frogger im TigerJython-Skript (S. 229 ff.).
In der OOP sind insbesondere die Events **die** Schnittstelle zum Benutzer (User).
Events werden zum Großteil durch Nutzeraktionen ausgelöst. Ein Tastendruck, eine Mausbewegung oder auch ein Klicken mit der Maus sind beispielsweise durch den Nutzer (User) ausgelöste Events. Diese Events werden normalerweise vom Betriebssystem als der Überwacher aller Ein- und Ausgaben abgefangen und an das laufende Programm weitergegeben, in dessen Programmfenster dieser Event auftrat.
TigerJython ist über die die Klasse GameGrid mit diesen Events verbunden. Tritt zur Laufzeit des eigenen Programms ein bestimmter Event auf, so möchte man als Programmierer selbst darauf reagieren. Dann muss normalerweise ein sogenannter **Eventhandler** registriert (angemeldet) werden. Nach dieser Registrierung werden auftretende Events an diesen Eventhandler weitergeleitet und können dort programmtechnisch im Sinne der Applikation (des Programmes) behandelt werden.

Parameter des makegrid Konstruktors

In TigerJython unter Benutzung der *gamegrid* Bibliothek (Modul) sieht der Ablauf für eine **Eventregistrierung** mit Hilfe des Eventhandler folgendermaßen aus:

- Das Hauptprogramm (Main) behandelt (bis auf weiteres) alle Events.
- Der eigene Eventhandler (unterstrichen dargestellt) wird bei der Instanziierung eines Objektes der Klasse GameGrid „angemeldet“und somit im Konstruktor übergeben beim Erstellen des Objektes aus der Klasse. Der Name des eigenen Eventhandlers kann frei gewählt werden! Im unten angeführten Beispiel reagiert ein noch zu definierender Eventhandler mit Namen *onKeyRepeated* auf ein (Betriebssystem-)Ereignis des Typs *keyRepeated* (vgl. Dokumentation zu GameGrid):

```
myGrid = makeGameGrid(800, 600, 1, None, keyRepeated = onKeyRepeated)
```

In diesem Beispiel wird ein **GameGrid-Objekt** mit Namen *myGrid* erzeugt. Wie in der Dokumentation zur Klasse GameGrid nachzulesen ist, bekommt der Konstruktor mehrere Parameter:

- 800, 600 sind die Breite und Höhe des Grids in **der Anzahl von GameGrid-Zellen**.
- 1 ist die Größe einer (quadratischen) GameGrid-Zelle. In diesem Fall ist die GameGrid-Zelle somit genau ein Pixel groß) Somit besteht das Spielfeld aus 800 Feldern in der Breite und 600 Feldern in der Höhe mit der Größe von 1x1 Pixel pro Feld.
  Wählen wir hier statt der *1* eine *5* für die **Zellgröße,** bestünde das Fenster aus 800 Feldern in der Breite mit der Größe von 5 x 5 Pixeln pro Feld. Also ein 4000 Pixel breites Fenster!
  Dieses gilt für die Höhe analog: Im Beispiel wäre das Fenster somit 600 x 5 = 3000 Pixel hoch!
  Dieser Zusammenhang ist wichtig zu verstehen, wenn ein Spielfeld entworfen werden soll.
  Andersherum ausgedrückt: um ein Spielfeld mit der Pixelgröße von 800 x 600 zu erstellen und darin Felder der Größe 10 zu haben, müssen die ersten drei Parameter 80, 60, 10 lauten!
  (**Übung/Ausprobieren**: Welche Parameter müssen eingegeben werden, um ein 300 x 200 großes Spielfeld mit Feldern der Größe 5 (5 x 5 Pixel) zu bekommen?)
- Der Nullpunkt (0,0) ist **links oben**. Die X-Koordinate verläuft in horizontaler Richtung (hier bis 800)
- *onKeyRepeated* ist im o.a. Beispiel der **eigene** Eventhandler. Dieser wird im Hauptprogramm **als Funktion definiert** und im Konstruktor nur über den Namen mit dem festen (eingebautem) Event *keyRepeated* verknüpft.
- Um Tastatur-Events zu erfassen, können statt *keyRepeated* auch die anderen Events (Callbacks in der TigerJython-Dokumentation genannt) *keyPressed*(e) und *keyReleased*(e) registriert werden. Im Unterschied zu *keyRepeated(code)* muss der Keycode aber mit *e.getKeyCode()* aus dem Parameter e geholt werden. Zudem ist in diesem Spiel *keyPressed(e)* weniger geeignet, da es nach dem Drücken und Halten der Taste eine Verzögerung gibt, bis die nachfolgenden Press-Events ausgelöst werden.

Zum Beispiel erzeugt der Aufruf von

makeGameGrid(spalten, zeilen, groesse, False, keyPressed = onKeyPressed, mousePressed = pressCallback)

*ein Spiel*fenster mit *spalten* x *zeilen* Zellen der Größe *groesse* mit *schwarzer* Zellumrandung. Die Tastenevents und Mausevents werden registriert bei der Funktion *onKeyPressed und pressCallback. Die Funktionen können auch anders benannt werden, müssen nur mit dem gleichen Namen im Programm definiert werden:*

KeyCodes

Wenn die Keycodes nicht bekannt sind, so hilft ein kleines Testprogramm, das diese ausgibt:

```
from gamegrid import *

def onKeyPressed(event): # der Name des Parameters, hier event, kann frei gewählt werden
    print "Taste gedrückt, KeyCode lautet: ", event.getKeyCode()

def onKeyReleased(event):
    print "Taste losgelassen, KeyCode lautet: ", event.getKeyCode()

# main
makeGameGrid(800,600,1,None,"sprites/lane.gif", False,keyPressed = onKeyPressed, keyReleased = onKeyReleased)
show()
```

Weitere Hinweise zum Arbeiten mit GameGrid

Wenn ein GameGrid Fenster geöffnet erstellt wurde, kann **nur** dieses Fenster auch Tastendrücke annehmen! Es sollte somit immer den „Fokus“ haben.

Manchmal ist es sinnvoll, einen kleinen Statustext im Spielfenster zu integrieren.
GameGrid bietet dafür eine einfach Funktion setStatusText

addStatusBar(100) # erzeugt einen Textbereich im unteren Bereich des GameGrids der Größe 100 Pixel
setStatusText("Beispieltext.\nUmbruch mit dem Sonderzeichen \n, Font("SansSerif",Font.PLAIN,10), Color.black)

Font("SansSerif",Font.PLAIN,10), bedeutet die Auswahl des Schrifttyps, z.B. “Serif“, “Monospace“. Die zur Verfügung stehenden Schriftarten sind systemabhängig.

Font.PLAIN ist die Schriftart, z.B PLAIN für normale Schrift
Font.BOLD ist **fett** in der Darstellung.
Font.ITALIC stellt die *kursive* Darstellung.

(https://dbs.cs.uni-duesseldorf.de/lehre/docs/java/javabuch/html/k100155.html)

Color.black ist die Schriftfarbe. Dabei ist zu achten, dass die Farben ebenfalls systemspezifisch (und java-spezifisch sind). Es gibt auf jeden Fall die Grundfarben .Color.white. Color.yellow, etc.

(https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html)