TGM

Let’s play Nonogramm

5BHITM

Krickl | Seckin

17.01.2015

Let's play Nonogramm

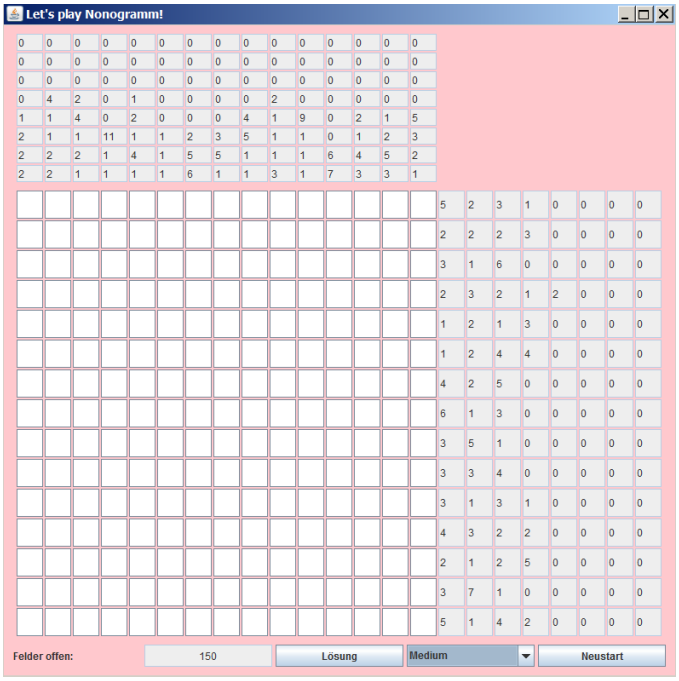
Inhalt

Aufgabenstellung 2
Zeitaufzeichnung 3
Allgemein 4
Wie startet man das Spiel 4
Verwendete Versionen 4
Sphinx Dokumentation 4
GUI 5
Probleme 5
Logik 5
Design 6
Quellen 8
Nachschlagen 8
Links vom Text 8

Aufgabenstellung

Nachdem Sie einige Designpattern in Python betrachtet haben, wollen wir uns einem wichtigen Entwurfmuster spielerisch nähern:

MVC



In einem Team (2) soll das Spiel Nonogramm umgesetzt werden.

- Spielfeld: 15 x 15
- Eine Statusleiste mit Anzeige der noch gesuchten Felder,
- Button zur sofortigen Lösung
- Button zum Neustart
- Auswahlfeld zur Einstellung der Schwierigkeit (EASY/200; MEDIUM/150; HARD/125; EXPERT/90; IMPOSSIBLE/50) auf Basis der gesuchten Felder!

Die Farbe rosa ist natürlich nicht Pflicht und könnte vielleicht vom User variabel eingestellt werden.

Viel Erfolg!

Ressourcen:

Unterlagen zu GUI-Programmierung in Python

https://de.wikipedia.org/wiki/Nonogramm

Zeitaufzeichnung

Task	Geschätzte Zeit in h	Tatsächliche Zeit in h	Verantwortung
Recherchieren und Nachschlagen	6	5	Krickl & Seckin
Algorithmus zum Generieren eines Nonogramms entwickeln	1	3	Krickl
GUI mit pyQt erstellen	1	3	Seckin
Import der GUI in PyCharm	0.3	1	Seckin
Zusammenfügen von View und Control	3	9	Krickl & Seckin
Actionhandling implementieren	2	1	Krickl & Seckin
Debuggen	2	3	Krickl & Seckin
Dokumentation des Codes	1	0.5	Krickl & Seckin
Protokoll erstellen	2	2	Krickl
Summe Krickl	15.3	14.25	
Summe Seckin	8.3	13.25	

Allgemein

Wie funktioniert Nonogramm?

„Das Spiel besteht aus einem Gitter aus beliebig vielen Kästchen. Ziel ist es, die Zellen eines Gitters so einzufärben (bzw. nicht einzufärben), dass die eingefärbten Kästchen in jeder Zeile und Spalte der dafür angegebenen Anzahl und Gliederung entsprechen. Die Zahlenfolge „4 2 1“ vor einer Zeile enthält beispielsweise die Information, dass in dieser Zeile (mit mindestens einem Kästchen Abstand) ein Block von vier zusammenhängenden Zellen, ein Block von zwei zusammenhängenden Zellen sowie eine einzelne Zelle in dieser Reihenfolge einzufärben sind. Aus der Kombination von Zeilen- und Spaltenangaben lässt sich eine (meist eindeutige) Lösung logisch herleiten.“ [1]

Wie startet man das Spiel

Man startet das Spiel indem man die Control.py Datei ausführt.
Sinn des Spiels ist alle offenen Felder (Anzahl in dem Textfeld rechts oben) blau zu bekommen.

Verwendete Versionen

Python 3.4.1.
PyQt 4.11.3

Sphinx Dokumentation

Sphinx im Pluginmanager installieren: Unter File -> Settings -> Project Interpreter -> + -> Sphinx

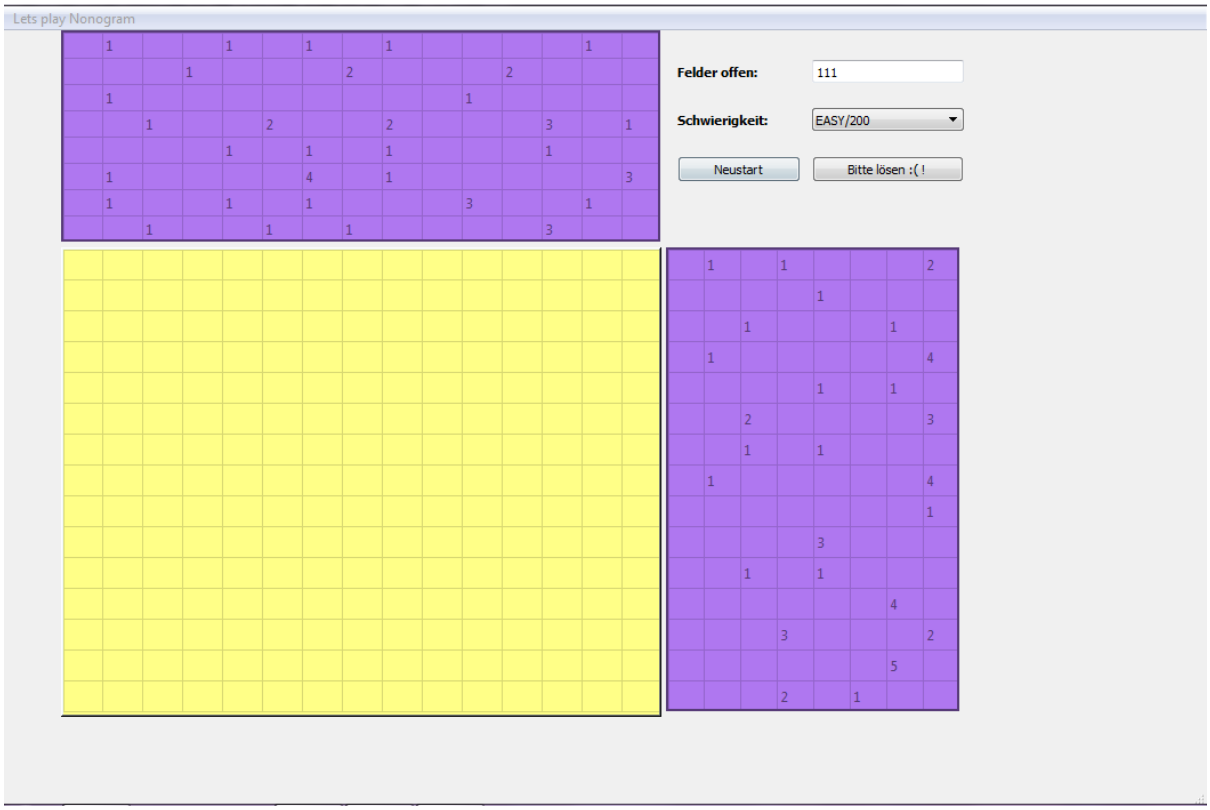
Sphinx einrichten: Unter Tools -> Sphinx quickstart ausführen.

Sphinx in das Projekt integrieren: Unter „Edit Confiurations“ (rechts oben oder unter Run) auf das + klicken. In der Dropdown Liste python docs – Sqhinx task auswählen. Beim Konfigurieren dieses Tools als Input den Ordner wo die index.rst Datei liegt wählen und als output ..project/doc as HTML wählen und oben noch einen Namen einfügen. Wenn man nun ein Kommentar innerhalb einer Methode mit 3“ anlegt, dann werden automatisch die Parameter und der Return tag eingefügt, diese muss man noch mit Inhalt befüllen.

Sphinx doc updaten: Im Projektfenster oben rechts durch das Drücken des grünen Pfeils.

GUI

In dem Spiel gibt es zwei Zustände jedes einzelnen Spielfelds: Blau oder gelb.
Dies wurde grafisch folgendermaßen umgesetzt:



Die Hintergrundfarbe wurde grau gewählt, damit es mit dem lila Hilfespalten ruhiger wirkt. Um die Spielfläche jedoch dezent abzuheben wurde diese gelb gemacht. Man kann die Felder markieren, das markierte Feld wird blau dargestellt.

Des Weiteren wurden die Menüpunkte in die leere Ecke rechts oben gegeben um eine gute Übersicht zu schaffen.

Die Schwierigkeitsstufen kann man in einem Dropdown Menü auswählen.

Probleme

Logik

Man findet sehr viele Algorithmen zur Lösung von Nonogrammen, jedoch aber wenige Ansätze zum Generieren. Ein Ansatz war ein 8-Bit Bild zu nehmen und aus dem ein Nonogramm zu generieren. Die von mir gewählte einfachere Methode funktioniert mit einer zufälligen Befüllung der Felder und nachträgliche Auszählung. [2]

Da man bei der Generierung das Spielfeld einfach speichern kann, braucht man keinen Lösungsalgorithmus und vergleicht das aktuelle Spielfeld mit dem Lösungsspielfeld.

Um diese Felder darzustellen verwenden wir ein 2D-Array. Dieses wird Standardmäßig mit doppelten Klammern angegeben und es mit `.append()` befüllen

```
arVar = [[ ]]
arVar[1].append( round(random() )
```

Doch da das nicht so funktioniert hat, habe ich das Array und die Befüllung folgendermaßen umgeändert [3]

```
arVar2 = [[0 for x in range(breite)] for x in range(breite)]
# schleife
arVar2[x][y] = round( random() )
```

Das Auszählen des Spielfelds wurde umgesetzt jedoch nicht fehlerfrei. Da in der GUI nur begrenzt Platz ist müsste das berücksichtigt werden, was nicht der Fall ist. Desweiteren muss man die einzelnen inneren Arrays des Tipps Arrays sortieren und die Nuller aussortieren um eine geeignete Darstellungsform zu erreichen, was auch noch nicht umgesetzt ist. Daher sind die Zahlen die am Rand angezeigt werden nicht falsch, jedoch unvollständig und keine Hilfe zum Lösen des Spiels.

Design

Zur Umsetzung und Verbindung der grafischen Komponente mit der Logik haben wir MVC verwendet. Jedoch stellte uns das vor einige Probleme, da viele Sachen nicht wie im alt bekannten Java funktioniert haben.

Die Verbindung zwischen den einzelnen Klassen sowie die Imports von den PyQt Klassen, haben mehr Probleme bereitet als gedacht.

In der Ui_MainWindow() Klasse importiert man

```
from PyQt4 import QtCore, QtGui
import sys
```

Und diese erbt folgendermaßen

```
class Ui_MainWindow(QtGui.QWidget):
```

Man startet die GUI so

```
Class Spiel(QtGui.QWidget):
def __init__(self):
        QtGui.QMainWindow.__init__(self)

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

#START:
if __name__ == '__main__':
    app = QtGui.QApplication(sys.argv)
    ex = Spiel()
    ex.show()
    sys.exit(app.exec_())
```

Durch die `self.ui` Variable kann man nun auf alle GUI Komponenten (Tabellen, Buttons, Textfelder etc.) zugreifen und diese auch während der Laufzeit ändern.

Die Variablen die man verwendet, werden im Konstruktor angelegt:

```
__init__(self):
    self.var1 = 0
```

Die Tabelle (QTable) muss nach dem erstellen mit QTableItem s befüllt werden. Die Items kann man dann färben bzw mit Text füllen und ihnen eine Ereignissteuerung hinzufügen.

Nachdem wir die Felder nur eingefärbt haben muss man jedem Item seinen Zustand (blau oder gelb) mitgeben. Dies wurde mit folgender Methode realisiert

```
self.ui.tableWidget.item(row, column).setWhatsThis(_fromUtf8("1"))
```

Aufrufen dieses Zustands funktioniert mit

```
zustand = self.ui.tableWidget.item(row, column).whatsThis()
```

Um Buttons eine Ereignissteuerung hinzuzufügen muss man ihnen ein ‚Callable Signal' mitgeben

```
self.pushButton_2.clicked.connect(Control.losen)
```

Um die Ereignissteuerung zu realisieren wird eine Methode losen() aufgerufen, jedoch aber ohne Klammern weil es Signal ist. Um weitere Parameter hinzuzufügen müssen entsprechend andere connect Methoden verwendet werden.

Die losen() Methode könnte wie folgt aussehen

```
def losen():
    print(„button wurde gedruckt")
```

Nun wird beim Drücken des Buttons auf der Konsole der angegebene String ausgegeben.

Um durch einen Buttonklick ein anderes GUI Element zu beeinflussen muss man dieses einfach aufrufen:

```
def losen():
    self.ui.lineEdit.setText(„neuer Text nach Buttonklick")
```

So wird beim Klicken der Text des Textfeldes dementsprechend umgeändert.

Die Änderung der Schwierigkeitsstufen wird im Hintergrund berücksichtigt und auch erfolgreich abgefragt jedoch wurde die Funktion nur im Control berücksichtigt, aber nicht in der GUI umgesetzt.

Quellen

Kompletter Code siehe GitHub
https://github.com/akrickl-tgm/nonogramm.git

Nachschlagen

Codeacademy – Learning Python
http://www.codecademy.com/

Algorithmus Ideen und Hilfe
http://www.entwickler-ecke.de/topic_NonogrammGriddler++Solver_60215,0.html

http://sourceforge.net/p/freenono/tickets/milestone/FreeNono%201.0/

PyQt4 API
http://pyqt.sourceforge.net/Docs/PyQt4/qtablewidgetitem.html

http://pyqt.sourceforge.net/Docs/PyQt4/qcombobox.html

http://pyqt.sourceforge.net/Docs/PyQt4/qtextline.html

Links vom Text

[1] **Nonogramm**
URL: http://de.wikipedia.org/wiki/Nonogramm
aufgerufen am 19.1.2015

[2] **Create a Nonogramm Puzzle**
http://codegolf.stackexchange.com/questions/30081/create-a-nonogram-puzzle
aufgerufen am 17.1.2015

[3] **How to define two dimensional array in python**
http://stackoverflow.com/questions/6667201/how-to-define-two-dimensional-array-in-python
aufgerufen am 18.1.2015