1 Textadventure 'Ravenswood Manor'

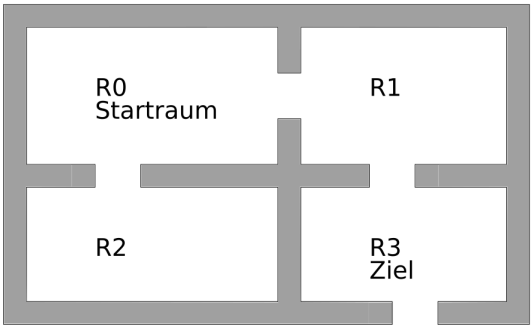
1.1 Was ist das, ein Textadventure?

Die berühmtesten Textadventures sind die `Zork`-Adventure vom Ende der 70er Jahren des zwanzigsten Jahrhunderts. Kommerziell waren sie Anfang der 80er Jahre sehr erfolgreich. Heutzutage wird an Stelle von Textadventure gerne der Begriff „interactive fiction“ verwendet. In einem Textadventure gibt das Spiel seine Beschreibungen und Reaktionen als Text aus und der Spieler gibt einfache Anweisungen als Text an das Spiel. Dies ist eine gute Grundlage, um in der Einfachheit gleichzeitig auch die Herausforderungen eines eigenen Spiels zu erleben. Im Fall des vorgegebenen Grundgerüsts von Ravenswood Manor kann das Spiel bisher die Richtungen `go north`, `go south`, `go east` und `go west` verstehen. Außerdem gibt es die Kommandos `help` und `quit`.
Letztlich findet das Meiste des Spiels im Kopf des Spielers statt und es braucht gute Ideen, da man den Spieler nicht mit bombastischen grafischen Effekten blenden kann. Klassisches Setting ist das Erkunden unterirdischer Labyrinthe oder hier bei uns eines Hauses. Dabei kann der Spieler Räume erkunden und dort Dinge finden, mitnehmen und anwenden. Er muss Rätsel lösen und Fallen entkommen.

1.2 Ravenswood Manor

`https://replit.com/@makerspacewut/textadventure-starter-ravenswood` enthält ein Grundgerüst für den einfachen Start. Am Anfang besteht das Haus nur aus vier Räumen. Jetzt möchte das Haus von dir erweitert werden!

Das Haus am Anfang



1.3 Die Herkunft des Spiels

Die ursprüngliche Fassung dieses Spiels wurde in der ersten Ausgabe des Magazins „Hello World“ der Raspberry Pi Foundation veröffentlicht. Es lässt sich grundsätzlich spielen. Man kann allerdings noch nicht wirklich viel machen.

1.4 Du und deine Aufgabe

Als Anwendungsentwicklerin magst du lieber programmieren als spielen und freust dich, wenn du der Gott deiner kleinen Welt bist. Hier hast du tatsächlich die volle Kontrolle über alles, darfst alles bestimmen, alle Regeln nach deinen Wünschen festlegen und alles gestalten, so wie du es gut und richtig findest.
Deine Aufgabe ist es also, das Spiel zu erweitern. Dabei darfst du deiner Kreativität freien Lauf lassen.

1.5 Erläuterungen zum Quelltext

`allowed_commands` Diese Liste enthält die erlaubten Worte, die das Spiel versteht:
`['go north', 'go south', 'go east', 'go west', 'help', 'quit']`

- Wenn du die neuen Befehle `'go up'` und `'go down'` einführen willst, musst du sie der Liste `allowed_commands` hinzufügen.
- Du musst die neuen Dictionaries `up` und `down` für die möglichen Bewegungen des Spielers anlegen. Auch alle bereits bestehenden Dictionaries müssen angepasst und erweitert werden.
- Als drittes musst du den `compass` anpassen.
- Als vorerst letztes kann und sollte man die Raumbeschreibungen anpassen. Das Spiel lebt von guten Beschreibungen.

Im folgenden Codefragment wird ein neuer Raum `R4` oberhalb von Raum `R2` eingefügt. Wobei sich *oberhalb* nur durch die Befehlswörter `go up` und `go down` ergibt.

```
north = { 'R0':None, 'R1':None, 'R2':'R0', 'R3':'R1', 'R4':None }
south = { 'R0':'R2', 'R1':'R3', 'R2':None, 'R3':None, 'R4':None }
east =  { 'R0':'R1', 'R1':None, 'R2':None, 'R3':None, 'R4':None }
west =  { 'R0':None, 'R1':'R0', 'R2':None, 'R3':None, 'R4':None }

up =    { 'R0':None, 'R1':None, 'R2':'R4', 'R3':None, 'R4':None }
down =  { 'R0':None, 'R1':None, 'R2':None, 'R3':None, 'R4':'R2' }

allowed_commands = ['go north','go south','go east', 'go west',
              'help', 'quit', 'go up', 'go down']

compass = { 'go north': north, 'go south': south, 'go east': east,
   'go west': west, 'go up':up, 'go down':down } # Zugriff auf up/down

description = {
  'R0': 'Du bist im Kaminzimmer. Eine Zeitung liegt auf dem Tisch.',
  'R1': 'Eine Art Bibliothek mit einem lila Sofa auf der Nordseite.',
  'R2': '''Die Küche ist gigantisch groß.
Auf dem Herd steht ein Topf groß wie eine Badewanne.''',
  'R3': 'Die Diele: Du siehst die Ausgangstüre.',
  'R4': 'Du bist in einer kleinen Dachkammer ohne Fenster.'  # neu!
}
```

`compass` ist ein Dictionary von Dictionaries. Es erfolgt also ein doppelt indizierter Zugriff über die vorgegebenen vier Dictionaries mit den Himmelsrichtungen. Doppelt indiziert heißt nur, dass der Python-Interpreter der Reihe nach in zwei Schritten die Variablen einsetzt und das Ergebnis berechnet.

Don't PANIC!

Ich bin in Raum `'R0'`. Dort gebe ich als Richtung `'go south'` an.

Der Aufruf von `compass[command]` liefert dann `compass['go south']`, und dass ist dann das Dictionary `'south'`. Dieses wird dann als `south['R0']` aufgerufen und liefert `'R2'` zurück. `'R2'` wird dann als `current_room` gesetzt und ich bin vom Raum `'R0'` in den Raum `'R2'` gegangen. Ganz einfach.

`north, south, east, west` sind vier Dictionaries, die die Karte des Hauses in Computerform darstellen. In den Dictionaries sind die verbotenen Richtungen durch das Python-Wort `None` gekennzeichnet. Erlaubte Richtungen hingegen sind als <Startraum>:<Zielraum> angegeben.

Es bietet sich bei der Erweiterung an, alle Räume *sortiert* in diesen Dictionaries einzutragen, da man sonst später bei fehlenden Räumen bösen Fehlern begegnet. Diese sind erfahrungsgemäß schwer zu finden!

Hauptschleife Die zentrale Hauptschleife des Programms ist als `while`-Schleife realisiert. Solange `current_room is not None` gilt, ist man im Haus und kann etwas tun. Wobei genau genommen das Programm läuft und das tut, was du programmiert hast.

Benutzereingabe Innerhalb der Hauptschleife wird in einer weiteren `while`-Schleife die Benutzereingabe geprüft und nur gültige Eingaben werden akzeptiert.

1.6 Eine Auswahl weiterer möglicher Ideen

- Das Haus kann viel größer und vielschichtiger sein.
- Die Raumbeschreibung kann bunter, vielschichtiger, lustiger sein.
- Wenn man ein Kommando `'look'` einführt, kann dieses weitere Details über den jeweiligen Raum offenbaren, die über die normale Beschreibung hinaus gehen. `'look'` ist keine Bewegung, muss also nicht in `compass` eingetragen werden, wohl aber in `allowed_commands`
- Alle Rätsel müssen in den Beschreibungen genügend Hinweise haben, um lösbar zu sein. Auch sollte es subtile Hinweise auf Fallen geben.
- Eine mögliche weitere Funktion wäre `'take'`, das Mitnehmen von Dingen. Diese Dingen sollten vom Spiel verwaltet werden. Das braucht ein Inventar. Dieses Inventar kann als Dictionary realisiert werden.
- Die Dinge im Inventar wollen auch zurück gelegt oder noch lieber verwendet werden: `'use sword on door'`.
- Das Mitnehmen kann gefährlich sein, wenn es Fallen auslöst. Das geht mit einer weiteren Datenstruktur, in der die Eigenschaft „Falle“ vermerkt ist. `'The door explodes. You are dead.'`
- Prinzen, Frösche, goldene Bananen und je nach Spielercharakter unterschiedliche Questen.
- Mit den verwendeten Datenstrukturen kannst du das Haus und seine Eigenschaften zur Laufzeit verändern: Wege können entstehen oder verschwinden, gruselige Dinge geschehen nur zu bestimmten Zeiten oder wenn man aus bestimmten Richtungen kommt, ...

 © 2022 Roland Härter

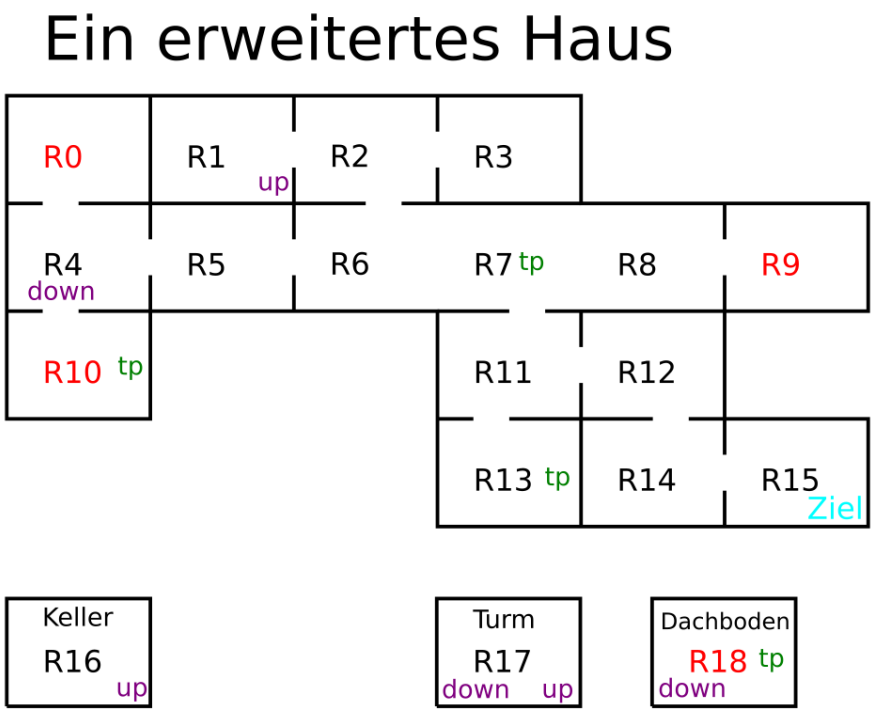
1.7 Eine erweiterte Hauptschleife

Hier zur Inspiration eine bereits ein wenig erweiterte Hauptschleife:

```
# ------------------------------------------------------------
command = ''
hilfe()    # Kurze Anleitung ausgeben
# Keep asking them which direction to go in
while( current_room is not None ):

   # Describe the current room
   # Access the dictionary with dictionary_name[lookup_value]
   print ( description[current_room] )

   # Ask what they want to do and validate it - only allowed_commands
   command = input('What do you want to do? ').lower()
   while command not in allowed_commands:
      command = input('What do you want to do? ').lower()
   if command == 'help':
      hilfe()
   elif command == 'quit':
      quit()     #  das Spiel freiwillig aufgeben :-/
      current_room = None
   elif command == 'look':
      print ( look_around[current_room] )
   elif ( command == 'take' ):
      if ( take(current_room) ):
         current_room = None     # you managed to die :-)
   # Look up whether a path that way exists and if so, go to that room
   elif compass[command][current_room] is not None:

      current_room = compass[command][current_room]

      # See if they are in the final room
      if current_room == final_room:
         result = input ('Willst du das Haus verlassen? [yes/No] ').lower()
         if result in ['y', 'yes', 'j', 'ja', 'mach schnell jetzt']:
            current_room = None # Ends the game loop
            print('You have found the exit. Finally.')
         else:
            continue
   else:
      print ('There is no path in that direction. ', end='')
```

1.8 Ein erweitertes Haus

Nach einer kreativen Phase als Architekt und Baumeister hat sich eine erweiterte Version des Hauses gebildet. In diesem Haus gibt es einen Keller und eine Bibliothek im Turm und darüber noch einen Dachboden. Um hoch und runter zu gelangen, braucht es passende Kommandos. Nur durch die Namen entsteht beim Spieler die Illusion davon, hoch und runter zu gelangen. Man kann auch Teleportation in das Spiel einbauen. Das ist hier im Beispiel bereits erfolgt.



Die Erweiterung des Hauses findet also zum einen über die neuen Kommandos `up`, `down` und `tp` statt. Zum anderen müssen die vorhandenen Dictionaries erweitert werden und es braucht neue Dictionaries mit den hinzugefügten Richtungen.
In der Erweiterung sind auch zufällige Starträume eingebaut, damit es abwechslungsreich ist und der Spielspaß länger anhält. Die möglichen Starträume sind im Bild in rot beschriftet.

Hier folgt nun die für den Computer verständliche Form dieses Hauses.

```
allowed_commands = ['quit', 'w', 'a', 's', 'd', 'tp',
   'north', 'south', 'east', 'west', 'up', 'down', 'look', 'take']

# Die WASD-Steuerung nutzt die gleichen Dictionaries wie
# die ursrünglichen Richtungs-Kommandos
compass = { 'w': north, 'a': west, 's': south, 'd': east,
  'go north': north, 'go south': south, 'go east': east, 'go west': west,
  'go up':up, 'go down':down, 'tp':tp }

# Die Dictionaries sind hier aus Platzmangel ohne Leerzeichen geschrieben
north={'R0':None,'R1':None,'R2':None,'R3':None,'R4':'R0','R5':None,\
'R6':'R2','R7':None,'R8':None,'R9':None,'R10':'R4','R11':'R7','R12':'R8',\
'R13':'R11','R14':'R12','R15':None,'R16':None,'R17':None,'R18':None}

south={'R0':'R4','R1':None,'R2':'R6','R3':None,'R4':'R10','R5':None,\
'R6':None,'R7':'R11','R8':None,'R9':None,'R10':None,'R11':'R13','R12':'R14',\
'R13':None,'R14':None,'R15':None,'R16':None,'R17':None,'R18':None}

east={'R0':None,'R1':'R2','R2':'R3','R3':None,'R4':'R5','R5':'R6',\
'R6':'R7','R7':'R8','R8':'R9','R9':None,'R10':None,'R11':'R12','R12':None,\
'R13':None,'R14':'R15','R15':None,'R16':None,'R17':None,'R18':None}

west={'R0':None,'R1':None,'R2':'R1','R3':'R2','R4':None,'R5':'R4',\
'R6':'R5','R7':'R6','R8':'R7','R9':'R8','R10':None,'R11':None,'R12':'R11',\
'R13':None,'R14':None,'R15':'R14','R16':None,'R17':None,'R18':None}

tp={'R0':None,'R1':None,'R2':None,'R3':None,'R4':None,'R5':None,'R6':None,\
'R7':'R13','R8':None,'R9':None,'R10':'R18','R11':None,'R12':None,\
'R13':'R10','R14':None,'R15':None,'R16':None,'R17':None,'R18':'R7'}

up={'R0':None,'R1':'R17','R2':None,'R3':None,'R4':None,'R5':None,\
'R6':None,'R7':None,'R8':None,'R9':None,'R10':None,'R11':None,'R12':None,\
'R13':None,'R14':None,'R15':None,'R16':'R4','R17':'R18','R18':None}

down={'R0':None,'R1':None,'R2':None,'R3':None,'R4':'R16','R5':None,\
'R6':None,'R7':None,'R8':None,'R9':None,'R10':None,'R11':None,'R12':None,
'R13':None,'R14':None,'R15':None,'R16':None,'R17':'R1','R18':'R17'}
```