Snake

1

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Snake Namespace Reference

Classes

- class Game

Functions

- def createfield ()
- def firststart ()

Variables

- **s** = socket.socket()
- int **size** = 25
- bool **feedcollision** = False
- int **length** = 1
- **direction** = randint(1,4)
- int **xx** = 8
- int **yy** = 8
- list **feedpoint** = [randint(1,16),randint(1,16)]
- list **xcoordinate** = [xx]
- list **ycoordinate** = [yy]
- bool **lost** = False
- int **score** = 0
- int **highscore** = 0
- **game** = Game()

4.1.1 Detailed Description

```
Informatik Projekt 2018
Thema:  Snake
Erstellt von: Adrian Bergen(4228620), Lukas Hansen(4463302), Fanziska Bollhorst (4463145)

Am Anfang werden die benötigten librarys importiert. Tkinter für die GUI,
Socket für die Kommunikation, Random für zufällige Zahlen.

Außerdem werden die für das Spiel benötigten Anfangsvariabeln erstellt.
size: Größe der Quadrate, auf den das Spielfeld gezeichnet wird.
feedcollision: Am Anfang kollidiert die Schlange nicht mit einem Punkt
length: Länge der Schlange, vergrößert sich beim Aufsammeln eines Punktes
direction: Zahlen von 1-4, sie stehen für die jeweilige Richtung.
Dabei wird direction von Arduino gesteuert.
xx und yy: Reine Anfangskoordinaten der Schlange auf dem Feld.
(Das Feld ist 16x16)
x/ycoordinate: xx und yy werden in jeweils einem Array gepackt.
Später zeigt das Array die Punkte aller Schlangenpunkte.
(Also länge des Arrays =Länge der Schlange)
lost: Ist standartmäßig nicht True
score/ Highscore: Ist am Anfang 0
```

4.1.2 Function Documentation

4.1.2.1 createfield()

```
def Snake.createfield ( )
```

```
Hier wird das erste mal das richtige Spielfeld erstellt. Es wird der Spielername übernommen
und nach oben links gesetzt, Score und Highscore nach oben rechts. Das Startfenster wird gelöscht.
Der Ticker wird zum ersten mal gestartet. Der Feedpoint wird gesetzt.
```

4.1.2.2 firststart()

```
def Snake.firststart ( )
```

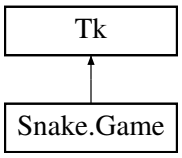
```
Wird das Spiel gestartet, erscheint ein Fenster mit Namenseingabe und einem Startknopf.
Durch diesen Startknopf startet sich das Spiel.
```

Chapter 5

Class Documentation

5.1 Snake.Game Class Reference

Inheritance diagram for Snake.Game:



Public Member Functions

- def createbutton (self)
- def ticker (self)
- def drawsnake (self)
- def feedpoint (self)
- def createfeedpoint (self)
- def collisions (self)
- def doastep (self)
- def restart (self)
- def receivearduino (self)

5.1.1 Member Function Documentation

5.1.1.1 collisions()

```
def Snake.Game.collisions (
              self )
```

```
Diese Funktion checkt die Kollision mit den Spielfeldrändern und der Schlange selber.
Passiert dies, wird lost=True gesetzt und der Ticker läuft im nächsten
Durchgang nicht weiter.
```

5.1.1.2 createbutton()

```
def Snake.Game.createbutton (
              self )
```

Hier wird ein Knopf erstellt, nachdem man bei dem Spiel verliert.
Dieser Knopf dient dazu das Spiel neu zu starten (Aufrufen der Funktion "restart").
Des Weiteren wird ein Textfeld erstellt. Dieses benachrichtigt dich, dass du verloren hast.

5.1.1.3 createfeedpoint()

```
def Snake.Game.createfeedpoint (
              self )
```

Hier wird der feedpoint generiert. Es passiert zufällig auf dem Spielfeld. Es ist darauf
zu achten, dass der Punkt nicht auf der Schlange generiert wird, andernfalls wird ein
neuer Punkt generiert.

5.1.1.4 doastep()

```
def Snake.Game.doastep (
              self )
```

Bestimme die neuen Koordinaten von Snake, abhängig von der momentanten Richtung.

5.1.1.5 drawsnake()

```
def Snake.Game.drawsnake (
              self )
```

Zuerst wird geprüft, ob die Schlange mit einem Punkt (Futter) kollidiert.
Anschließend wird das Array x/ycoordinate um die jeweilige NEUE x/y Koordinate erweitert.
Bei collisions wird die Kollision mit der Schlange selber, sowie den Wänden geprüft.
doastep verändert xx und yy, jenachdem welche "direction" die Schlange gerade hat.
Als nächstes wird der neue Kopf der Schlange gezeichnet, anschließend wird
hinten wieder die Spielfeldfarbe gezeichnet - Eine Schlange entsteht.
Da die Länge der Schlange von Punktekollisionen gesteuert wird, muss sich das
Array darauf anpassen, so wächst die Schlange tatsächlich auch.

5.1.1.6 feedpoint()

```
def Snake.Game.feedpoint (
              self )
```

```
Kollidiert der Schlangenkopf mit den x und y Koordinaten des feedpoints?
Dann verändert sich der Score, die Schlange wächst um 1, ein neuer Punkt wird erstellt.
```

5.1.1.7 receivearduino()

```
def Snake.Game.receivearduino (
              self )
```

```
Empfange vom Arduino die Richtungsdaten (String). Man unterscheidet zwischen
'u', 're', 'l', 'o' für die jeweiligen Richtungen. Es ist darauf zu achten,
dass die Schlange keine 180° Wende machen kann.
```

5.1.1.8 restart()

```
def Snake.Game.restart (
              self )
```

```
Wenn man verliert, muss das Spiel neu aufgesetzt werden.
Das Spielfeld wird neu gezeichnet, ein neuer Highscore wird bestimmt.
Es erscheint ein Text, dass man verloren hat und die Variabeln werden
zurückgesetzt (z.B. die Koordinaten). Der Ticker wird neugestartet.
```

5.1.1.9 ticker()

```
def Snake.Game.ticker (
              self )
```

```
Das Herz des Spiels. Ticker ruft sich alle 0,6 Sekunden rekursiv mithilfe einer after-Funktion wieder auf.
Mit receivearduino wird die direction von Snake empfangen.
Die Funktion ruft sich nicht wieder auf, wenn man verloren hat. Stattdessen hört das Spiel auf und der
Neustartbutton wird erstellt.
```

The documentation for this class was generated from the following file:

- Snake.py