Dokumentation „snake“

Da das Programm 422 Zeilen lang ist wird die Funktionsweise des Programms aus Gründen der Länge nicht vollständig erklärt, stattdessen werden einfach die Resultate der einzelnen Funktionen nicht aber ihre Funktionsweise und die generelle Struktur, nicht aber die einzelnen Variablen gezeigt.

Für das Spiel werden 3 Imports benötigt: gturtle, entrydialog und random (gehört nicht zu den Tigerjython- sondern den Python-Modulen). Das gturtle-Modul ist hierbei das wichtigste, es umfasst nicht nur die Figuren und das Spielfenster an sich, sondern ermöglicht einem auch den zugriff auf die Tastatur des Spielers. Das entrydialog- und random-Modul sind eher für Nischenfunktionen im Spiel zuständig. Entrydialog ermöglicht einem zum Beispiel das erstellen von benutzerdefinierten Dialogfenstern während random einfach nur eine Zufallszahl mit Minimum und Maximum generieren kann.

Grundsätzlich besteht der Code aus drei Teilen: den „global properties“, den Funktionen und dem Spiel selbst.

In den „global properties“ befinden sich alle Werte und Objekte, die im ganzen code verfügbar sein müssen. Schon der Ordnung wegen, werden hier alle Turtles erstellt. Auch alle Eigenschaften (der snake, der Steine, des Spielfelds, des Spielers, etc...) werden hier definiert.

Die einzelnen Abschnitte sind so geordnet: zuerst werden die KeyCodes definiert, direkt darunter werden die konstanten Eigenschaften des Spiels definiert (also alle Werte die sich während den Runden nicht mehr ändern), etwas weiter unten befinden sich die veränderbaren Werte (dinge die sich während dem Spiel noch ändern), anschliessend werden die EntryItems und Panes erstellt (bei den EntryItems handelt es sich um Objekte der EntryItem-Klasse, sie enthält Knöpfe, Schieberegler und vieles mehr, die Panes ihrerseits sind Objekte der EntryPanes-Klasse und haben die Möglichkeit ein EntryItem darzustellen) und schlussendlich werden die 4 Turtles (frameTurtle, snakeTurtle, stoneTurtle und appleTurtle) erstellt. Deren Funktion besprechen wir sobald sie benutzt werden.

Das Programm enthält ohne play() & main() (sie zähle ich nicht zu den klassischen Funktionen dazu, wir behandeln sie in ihrem eigenen Abschnitt) zehn Funktionen: openOptions(), generateStones(numberOfStones), deleteStones(numberOfStones), drawMap (color), turn(), TouchesStone(PosX, PosY), touchesApple (PosX, PosY), shuffleApple (), snakeTurtleIsAlive(PosX, PosY) und waitForInputName(). OpenOptions nutzt die, in global properties erstellten, Panes um ein Optionsfenster zu öffnen, in dem man diverse dinge ändern kann. GenerateStones generiert eine gegebene Anzahl Steine (Max. 10) an zufälligen Positionen, dafür klont es die stoneTurtle. DeleteStones löscht alle Steine, wenn man ihm sagt um wie viele es sich handelt. DrawMap zeichnet mit der frameTurtle die Spielfeldgrenze und wartet darauf, dass der Spieler das Spiel startet. Turn holt den letzten Tastendruck ab und dreht die snake, falls das den regeln entspricht und die richtige taste gedrückt wurde. TouchesStone überprüft ob die gegebenen Koordinaten einen Stein berühren.

TouchesApple überprüft ob die gegebenen Koordinaten den Apfel berühren. ShuffleApple setzt den Apfel auf eine zufällige Position. snakeTurtleIsAlive gibt True oder False zurück je nachdem ob ein Stein berührt oder das Spielfeld verlassen wurde. WaitForInputName holt nach dem Tot mithilfe eines Dialogfensters einen Namen ab, falls ein neuer Highscore erzielt wurde.

Das Spiel selbst besteht aus einer Main- und einer Play-Funktion. Die Main-Funktion dient dem Programm als Startpunkt, sie wird als erstes aufgerufen und enthält das ganze Spiel, soweit die Theorie, in der Praxis ruft sie allerdings eigentlich nur drawMap() und anschliessend play() auf.

Die Play-Funktion ist eigentlich der Dreh- und Angelpunkt des Programms. In ihr finden alle wichtigen Prozesse statt. Als erstes setzt sie die snake in die Mitte des Spielfelds und den Apfel auf eine zufällige Position. Anschliessend werden die Punkte auf 0 gesetzt und unten in der Statusbar angezeigt. Gleichzeitig wird überprüft ob der Highscore grösser ist als 0 und falls das der fall ist ebenfalls angezeigt. Anschliessend wird der Parameter isAlive initiiert und als True definiert. Direkt danach folgt eine While-Schleife, die sich solange wiederholt bis isAlive nicht mehr True ist. In dieser schleife wird die snake erst einmal bewegt, dies geschieht mithilfe der turn() funktion und einem einfachen snakTurtle.forward(movementSpeed) das die snake um den Wert von movementSpeed in Pixel nach vorne bewegt. Daraufhin wird die Liste snakePos, in der die X & Y Koordinate der snake gespeichert sind, aktualisiert. Danach wird mit der snakeTurtleIsAlive-Funktion überprüft ob die snake noch lebt und falls das nicht der Fall ist wird isAlive auf False gesetzt. Als letztes wird in der Schleife mit der touchesApple-Funktion überprüft ob die snake den Apfel berührt, sollte dies der Fall sein wird ein hoher Ton gespielt, die Punkte um 100 erhöht, der Wert von movementSpeed mit dem Wert von msScaling addiert (sollte msScaling + movementSpeed mehr als 37 ergeben wird der Wert stattdessen auf 37 gesetzt) und die Geschwindigkeit der snakeTurtle auf das 10 fache des movementSpeed wertes gesetzt um die sichtbare Geschwindigkeit der Turtle auch wirklich zu erhöhen.

Sobald die snake stirbt und die Schleife verlassen wird, wird ein hoher ton gespielt, der Statustext auf «GAME OVER! Press Esc for options & Enter to play again» gesetzt und der neue Highscore (falls erreicht) sowie der dazugehörige Name mit waitForInputName() in den dazugehörigen globalen Variablen gespeichert. Anschliessend wird der Parameter waitingToRestart initiiert und als True definiert. Danach wird gleich wie Oben eine While-Schleife gestartet, die sich so lange wiederholt bis waitingToRestart nicht mehr True ist. In dieser wird der letzte Tastendruck abgeholt und es wird überprüft ob die Taste Esc oder Enter war. Falls der letzte Tastendruck Esc war, werden mit der deleteStones-Funktion alle alten Steine gelöscht, mit openOptions() das Optionsfenster geöffnet und mit der generateStones-Funktion neue Steine generiert. Unmittelbar darauf wird die snake wieder in die Mitte des Spielfelds, ihre Blickrichtung nach oben, movementSpeed auf den Anfangswert und waitingToRestart auf False gesetzt. Dann wird play() erneut aufgerufen.

Sollte der Tastendruck allerdings Enter sein, passiert eigentlich exakt dasselbe, nur wird nach dem löschen der alten Steine kein Optionsfenster geöffnet sondern es werden sofort neue generiert.