**Hangman**

Fach: Informatik

Verfasser: Noel Backhaus

Klasse: Jahrgangsstufe 1

Lehrer: Frau Deubler

Schuljahr: 2023 / 2024

Inhalt

[1. Aufgabe 1](#_Toc155280119)

[2. Schritte der Lösung 2](#_Toc155280120)

[3. Programm 5](#_Toc155280134)

[4. Fazit 8](#_Toc155280135)

[5. Quellen 9](#_Toc155280136)

[6. Versicherung 11](#_Toc155280137)

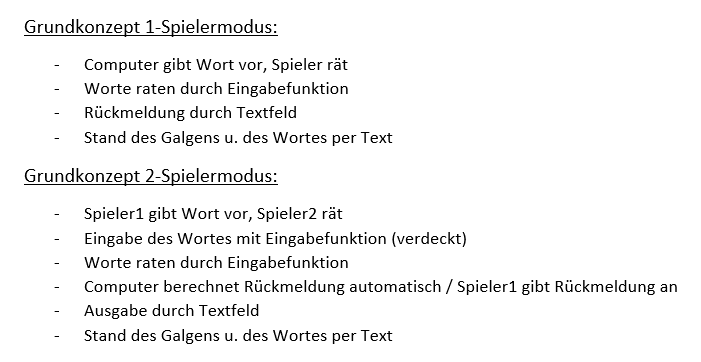
# Aufgabe

Als Aufgabe für meine GFS habe ich mir ausgesucht, das Spiel „Hangman“ auf dem Computer zu programmieren. Die Bedingungen waren, dass man ein englisches oder deutsches Wort auswählen kann, dass es einen Ein- und einen Zweispieler Modus gibt und dass eine Plausibilitätskontrolle stattfindet, also dass überprüft wird, ob die Eingabe korrekt ist. Ich habe mich entschieden, die Programmiersprache Python und die Entwicklungsumgebung IDLE zu benutzen.

# Schritte der Lösung

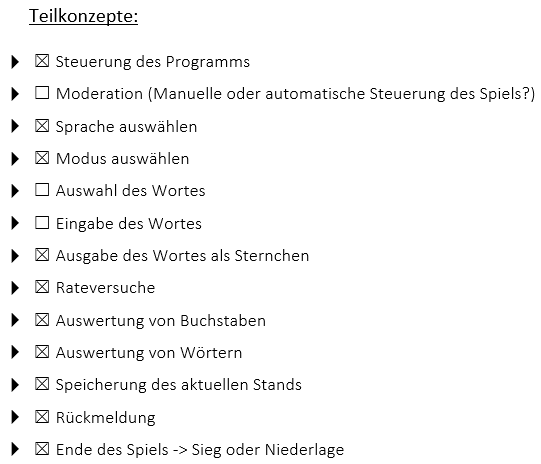
Grundkonzept

Der erste Schritt, um das Spiel zu programmieren, war, dass ich die Bedingungen festgelegt habe und Fragen geklärt habe und dann habe ich mir ein Grundkonzept für den Ablauf des Spiels aufgeschrieben. Ich habe jeweils eins für den Ein-Spielermodus und den Zwei-Spielermodus gemacht. Das sah so aus:



Teilkonzepte

Danach habe ich das Grundkonzept in Teilkonzepte aufgeteilt, in denen ich die einzelnen Schritte beschrieben habe, die ich brauche, um die einzelnen Teile des Programms fertigzustellen. Die einzelnen Teile habe ich in Stichpunkten aufgeschrieben, die dann noch in weitere Punkte aufgeteilt sind. Dabei habe ich versucht, jeden einzelnen Schritt aufzuschreiben, der für die Umsetzung eines Konzepts gebraucht wird. Im späteren Verlauf der Arbeit habe ich die Überschriftenfunktion in Word benutzt, um die Teilschritte auf- und zuklappen zu können. Außerdem habe ich Kontrollkästchen hinzugefügt, um abhaken zu können, welche Punkte ich erledigt habe. Ich habe hier nur ein Bild eingefügt, bei dem die Punkte zugeklappt sind, weil alle Punkte mit Unterpunkten zu viel Platz einnehmen würden. Die Teilkonzepte sind natürlich auch während dem Programmieren noch angepasst worden, aber so sahen sie am Ende aus:

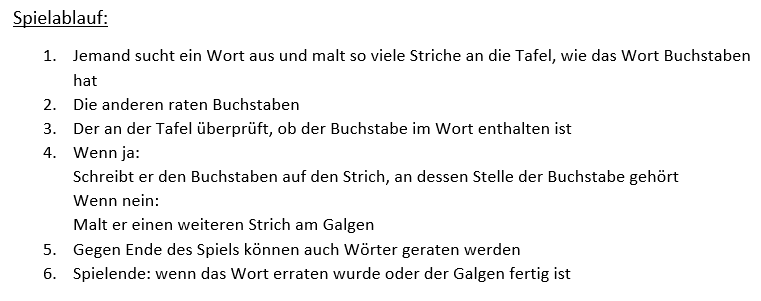


Programmierarbeit

Als Nächstes habe ich mir die einzelnen Teilkonzepte vorgenommen und angefangen, sie in Code umzusetzen. Dafür habe ich auf mein bisheriges Wissen zurückgegriffen und wenn ich etwas nicht wusste, wie z.B. wenn ich eine Funktion nicht kannte, habe ich im Internet gesucht oder ich habe mir selbst eine Lösung ausgedacht, indem ich in einer separaten Datei solange rumprobiert habe, bis der Programmausschnitt funktioniert hat. Später, als das Programm größer wurde, habe ich manchmal Struktogramme benutzt, um die Übersicht zu behalten.

Spielablauf

Zwischendurch habe ich mir nochmal separat den Spielablauf auf Papier geschrieben, um mein Spiel mit einer echten Runde Galgenmännchen vergleichen und um sehen zu können, in welcher Reihenfolge die Konzepte drankommen müssen, wie ich die einzelnen Schritte des Spiels am Computer umsetzen kann im Gegensatz zum Spielen an der Tafel oder auf Papier und was ich für mein Spiel ändern muss. Mein Plan des Spielablaufs sah so aus:



# Programm

Grundlage

Ich nehme an, dass jeder das Spiel „Hangman“ bzw. „Galgenmännchen“ kennt. Dabei muss man ein unbekanntes Wort mithilfe von Buchstaben erraten, wenn man falsch rät, wird Stück für Stück ein Galgen aufgezeichnet, an dem ein Strichmännchen hängt. In meinem Programm wird der Galgen allerdings durch Wörter dargestellt, da es rein textbasiert ist.

Aufteilung des Programms

Das Programm ist unterteilt in die Bereiche Variablen, Unterprogramme und das Hauptprogramm. Noch vor dem Bereich Variablen wird vom Modul „random“ die Funktion „randint“ importiert, weil sie für eine bestimmte Funktion benötigt wird und nur durch das Modul zugänglich ist. Im Bereich Variablen werden globale Variablen definiert, die im ganzen Programm genutzt werden können. Im Gegensatz zu globalen Variablen stehen lokale Variablen, die nur innerhalb der Funktion genutzt werden können, in der sie definiert sind. Im Bereich Unterprogramme sind die Unterfunktionen definiert, die die einzelnen Teile des Programms darstellen und die im Hauptprogramm nur noch nach und nach aufgerufen werden müssen. Durch das Hauptprogramm wird das Spiel gesteuert, da dort die Funktionen aufgerufen werden, die für den Spielablauf gebraucht werden. Das Hauptprogramm ist noch einmal unterteilt in die Vorbereitung und das eigentliche Spiel, weil nun mal einige Vorbereitungen für das Spiel getroffen werden müssen.

Funktionsweise des Programms

In der Vorbereitung des Spiels im Hauptprogramm werden die Sprache, der Spielmodus und das Wort ausgewählt oder das Wort wird von einem anderen Spieler eingegeben. Dann wird das Wort verschlüsselt und nachdem die Konsole geleert wurde, wird der aktuelle Stand zum ersten Mal ausgegeben und das Spiel geht los. Da es in der Entwicklungsumgebung IDLE keinen Befehl zum Leeren der Konsole gibt, muss man dafür eine große Anzahl an leeren Zeichenketten ausgeben, damit es so aussieht, als wäre die Konsole leer. Die Unterprogramme und Variablen sind mit sprechenden Bezeichnern benannt, das heißt, dass sie Namen haben, die genau das beschreiben, was sie auch tun, bzw. wofür sie da sind. Damit ist die Vorbereitung des Spiels abgeschlossen. Der nächste Teil des Hauptprogramms ist der Spielstart. Hierbei geht das eigentliche Spiel los und zwar wird mit einer while-Schleife geregelt, dass das Spiel solange läuft, bis das Wort erraten wurde oder der Galgen fertig ist. In jedem Durchgang wird ein Buchstabe oder ein Wort geraten. Diese werden ausgewertet, nachdem die Konsole geleert wurde, das Wort und der Galgen werden aktualisiert und als aktueller Stand ausgegeben.

In der Unterfunktion „spracheauswaehlen()“ wird die Sprache ausgewählt. Als Erstes soll die Sprache vom Spieler eingegeben werden. Danach kommt die Plausibilitätskontrolle. Dabei wird überprüft, ob die Eingabe den Sprachen Englisch oder Deutsch entspricht. Wenn nicht, wird der Spieler aufgefordert, eine neue Eingabe zu machen. Das wird solange wiederholt, bis die Eingabe stimmt und sorgt dafür, dass man die Sprachen eindeutig zuordnen kann.

In der Funktion „modusauswaehlen()“ wird der Spielmodus ausgewählt. Der Spieler soll eingeben, ob er einen Ein- oder Zweispieler Modus möchte. Es findet wieder eine Plausibilitätskontrolle statt, bei der überprüft wird, ob die Eingabe korrekt ist.

Die Funktion „wortauswaehlen()“ wählt ein zufälliges Wort aus einer Liste in einer Textdatei aus. Dazu wird eine zufällige Zahl mit der Funktion „randint()“ erzeugt und mit einer For-Schleife wird das Wort ausgewählt, das in der Zeile steht, die der Zahl entspricht. Dieses Prinzip wird für beide Sprachen angewendet mit unterschiedlichen Textdateien. Die Textdatei für deutsche Wörter hat knapp 240000 Wörter, die für englische 355000. Bei deutschen Wörtern müssen zusätzlich noch Sonderzeichen ersetzt werden, da in der Textdatei einige Wörter mit Sonderzeichen vorhanden sind, die das Programm nicht verarbeiten kann, bzw. weil bei Hangman sowieso keine Sonderzeichen im Wort sein sollen. Das Wort soll allerdings nur beim Einspieler Modus ausgewählt werden, da das Wort beim Zweispieler Modus von einem anderen Spieler eingegeben wird.

In der Funktion „worteingeben()“ wird das Wort beim Zweispieler Modus eingegeben. Dazu gibt es wieder eine Eingabe und eine Plausibilitätskontrolle, bei dieser muss diesmal aber nicht die richtige Schreibweise kontrolliert werden, sondern dass nur Buchstaben eingegeben werden. Das wird mit einer For-Schleife gelöst, die jedes Zeichen in der Eingabe einzeln überprüft. Wenn die Bedingung für ein Zeichen zutrifft, wird die Überprüfung fortgesetzt, wenn sie nicht zutrifft, wird sie abgebrochen, weil kein falsches Zeichen im Wort sein darf. Außerdem kommt noch die Kontrollvariable „korrekt“ dazu. Nach der Plausibilitätskontrolle müssen auch hier wieder die Sonderzeichen ersetzt werden.

Die Funktion „verschluesseln()“ ist die kürzeste Funktion und sorgt nur dafür, dass eine Variable erzeugt wird, in der das Wort mit Sternchen statt Buchstaben angezeigt wird, damit der Spieler das Wort nicht sehen kann.

Die Funktion „raten()“ lässt den Spieler einen Buchstaben oder ein Wort raten. Dafür gibt es wieder eine Eingabe und eine Plausibilitätskontrolle. Anschließend wird anhand der Länge der Eingabe unterschieden, ob ein Buchstabe oder ein Wort geraten wurde und die Eingabe in der entsprechenden Variable gespeichert.

In der Funktion „buchstabeauswerten()“ wird überprüft, ob der geratene Buchstabe schon mal geraten wurde und ob er im Wort enthalten ist. Der Buchstabe wird sowohl als Großbuchstabe, als auch als Kleinbuchstabe überprüft. Wenn der Buchstabe nicht darin ist, wird die Variable „galgenindex“, die den Stand des Galgens steuert, um eins erhöht.

Wenn ein Wort geraten wurde, wird in der Funktion „wortauswerten()“ überprüft, ob das geratene Wort stimmt oder nicht. Wenn das Wort stimmt, ist das Spiel vorbei.

In der Funktion „entschluesseln()“ wird das verschlüsselte Wort aktualisiert und die Buchstaben, die erraten wurden, an die Stelle der entsprechenden Sternchen gesetzt. Das wird mit einer For-Schleife umgesetzt, bei der für jeden Buchstaben im Wort überprüft wird, ob er dem erratenen Buchstaben entspricht. Wenn ja, dann wird er zu einer Hilfsvariable hinzugefügt, durch die das verschlüsselte Wort neu zusammengesetzt werden kann. Dabei wird unterschieden zwischen Großbuchstaben und Kleinbuchstaben. Wenn der zu überprüfende Buchstabe nicht der erratene Buchstabe ist, wird zur Hilfsvariable das Zeichen hinzugefügt, das im verschlüsselten Wort an der nächsten Stelle kommt. So können die Sternchen, die noch nicht erraten wurden und die Buchstaben, die schon erraten wurden, zum neuen Wort hinzugefügt werden. Um die nächste Stelle zu identifizieren, wird eine Zählervariable benutzt, die dann nur als Index hinter die Variable des verschlüsselten Wortes geschrieben werden muss. Anschließend wird die Hilfsvariable wieder in die Variable für das verschlüsselte Wort übertragen. Als Letztes wird überprüft, dass verschlüsselte Wort das gleiche ist wie das gesuchte Wort, denn dann wurden alle Buchstaben erraten und das Spiel ist zu Ende.

Die Funktion „galgen()“ speichert den aktuellen Stand des Galgens. Dafür wird die Variable „galgenindex“ benutzt, die in der Funktion „buchstabeauswerten()“ erhöht oder eben nicht erhöht wurde. Außerdem werden zwei Listen erstellt, eine, in der ein fertiger „Galgen“ gespeichert ist, also alle Elemente des Galgens als Wörter aufgelistet und eine, die den tatsächlichen Galgen darstellt und am Anfang leer ist. Dann muss nur noch dem unfertigen Galgen das nächste Element hinzugefügt werden, wenn der Galgen „weitergezeichnet“ werden soll. Das wird mit einer For-Schleife geregelt, in der für jede Stufe des Galgens ein Wort vom fertigen Galgen an den unfertigen Galgen angehängt wird. Dann wird die Variable galgenstand dem letzten Element des Galgens gleichgesetzt, weil immer nur das letzte Element ausgegeben werden soll. Schließlich wird noch kontrolliert, ob der Galgen fertig ist, damit das Spiel in dem Fall beendet werden kann.

In der Funktion „rueckmelden()“ wird der aktuelle Stand des Wortes und des Galgens ausgegeben, bis das Spiel zu Ende ist. Dann wird der endgültige Stand ausgegeben.

# Fazit

Selbstständig ein eigenes Spiel zu programmieren, war eine gute Erfahrung für mich, ich habe gelernt, dass es sehr viel Planung erfordert und sehr lange dauern kann. Es hat Spaß gemacht, aber ich habe auch gemerkt, dass man an seine Grenzen kommen kann. Das Programm hat für mich weniger einen praktischen Nutzen, weil es schon viele Hangman-Spiele im Internet gibt, aber es hat für mich den Nutzen, dass ich Erfahrung gesammelt habe und ein eigenes Projekt fertiggestellt habe. Durch die GFS habe ich gemerkt, dass es mir Spaß macht, auch größere Projekte zu programmieren und vielleicht kann ich das später in meinem Beruf nutzen oder als Hobby ausüben.

# Quellen

Internetquellen:

29.10.2023:

<https://www.python-lernen.de/input-nutzereingaben.htm>

30.10.2023:

<https://www.python-lernen.de/if-abfrage-python.htm>

<https://www.heise.de/tipps-tricks/gz-Dateien-oeffnet-so-klappt-s-9309533.html#GZ-Dateien%20%C3%B6ffnen>

01.11.2023:

<https://www.geeksforgeeks.org/with-statement-in-python/>

<https://www.w3schools.com/python/ref_func_open.asp>

02.11.2023:

<https://pythonspot.com/zufallszahlen/>

<https://github.com/davidak/wortliste>

<https://www.delftstack.com/de/howto/python/remove-certain-characters-from-string-python/>

<https://www.w3schools.com/python/python_lists.asp>

<https://www.delftstack.com/de/howto/python/how-to-read-specific-lines-from-a-file-in-python/>

08.11.2023:

<https://github.com/kkrypt0nn/wordlists/blob/main/wordlists/languages/english.txt>

<https://www.delftstack.com/de/howto/python/global-variables-in-python-and-how-to-change-them-from-a-function/>

30.11.2023:

<https://www.delftstack.com/de/howto/python/stop-a-for-loop-in-python/>

<https://www.w3schools.com/python/ref_string_strip.asp>

15.12.2023:

<https://www.delftstack.com/de/howto/python/python-string-contains-word/>

30.12.2023:

<https://www.python-lernen.de/listen-methoden.htm>

# Versicherung

Ich erkläre, dass ich die Arbeit selbstständig angefertigt und nur die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken, auch elektronischen Medien, entnommen sind, sind von mir durch Angabe der Quelle als Entlehnung kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, Bilder und andere visuelle Darstellungen.

Römerberg, 10.01.2024,

