**Die Sprache als Medium zur Steuerung**

Seit längerem ist die Menschheit damit beschäftigt mit Computer zu interagieren und gesprochene Sprache aufzunehmen und zu verarbeiten. Die Geschichte der Spracherkennung beginnt 1877 mit einer Innovation von Thomas Edisons Phonograph. Dieser ist das erste Gerät, welches Töne aufnehmen und diese dann wieder reproduzieren kann. Im Jahre 1936 wurde ein weiterer Meilenstein von einem Team von Ingenieuren des Unternehmen Bell Labs erreicht. Sie haben den sogenannten „Voder“ entwickelt, welcher versuchte eine menschliche Stimme zu erzeugen. 1952 hat ein Team von Bell Labs den „Audrey“ entwickelt, welcher von einer einzelnen Stimme gesprochene Ziffern erkannt hat. Zehn Jahre später 1962 veröffentlichte IBM auf der damaligen Weltausstellung die „Shoebox“-Maschine (Abbildung 1.0), die 16 englischsprachige Wörter verstehen konnte. Zu dieser Zeit war das ein großer Erfolg, wenn man bedenkt wie einfach Computer zu dieser Zeit noch waren.

Abbildung 1.0

In den 1970 er Jahren erreichte die Spracherkennung weitere bedeutsame Fortschritte, das Spracherkennungssystem „Harpyie“ von einer Behörde des US-Verteidigungsministeriums namens DARPA, konnte 1011 Wörter verstehen. 1996 war es möglich fließende Sprache zu erkennen bzw. sie zu verstehen mit der Innovation „MedSpeak“. In den nächsten Jahrzehnten kamen immer mehr Systeme auf den Markt die noch mehr Wörter verstehen konnten und in Alltagsgegenstände eingebettet waren. Zuletzt hat Amazon ihr Produkt „Echo“ veröffentlicht, welcher ein sprachgesteuerter Lautsprecher ist. Echo wird von Alexa, einem digitalen persönlichen Assistenten betrieben.

Abbildung 1.01

**Die Bereiche in denen sich Spracherkennung schon etabliert haben**

In einigen Teilbereichen hat sich die Spracherkennung bewährt und wird tatsächlich eingesetzt, zum einen Kommandosysteme, wie zum Beispiel Assistenzsysteme, welche einfache Befehle ausführen. Diese sind vorzufinden in Smartphones oder Autos, für eine digitale Erfassung von Notizen oder um Einstellungen zu treffen. Zum anderen Diktiersysteme, welche für besondere Berufsgruppen verwendet werden, wie zum Beispiel Rechtsanwälte oder Ärzte. Seit geraumer Zeit wurden ebenfalls Dialogsysteme entwickelt, welche meistens telefonisch zur Anwendung kommen, die einfache „Ja/Nein“ Ausführungen tätigen oder persönliche Assistenten.

**Die Problematik der Wahrnehmung der gesprochenen Sprache**

* Aussprache
* Mehrdeutigkeiten
* Umwelteinflüsse
* Wortgrenzen

Einerseits ist Spracherkennung schwierig, weil Personen gleiche Wörter unterschiedlich aussprechen und daher die Zeitspanne eines Wortes mit demselben Schalldruckpegel sehr divers aussehen kann. Im Fachjargon ist hier die Rede von „Intrasprechervariabilität. Wie man anhand den unten angeführten Abbildungen 1.1 und 1.2 sehen kann, ist die Lautstärke und die Geschwindigkeit der Aussprache des Wortes „Spiegel“ eine komplett andere. Daher ist es nicht von großer Relevanz Wörter und ganze Sätze 1:1 zu überprüfen.

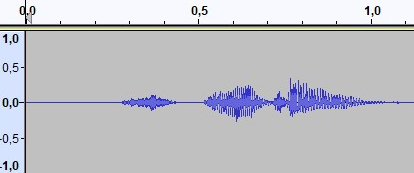
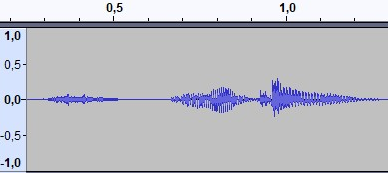


Abbildung 1.1 Abbildung 1.2

Andererseits ist Spracherkennung schwierig, wenn man den Faktor der fließenden Sprache miteinfließen lässt, da man jetzt nicht nur das gesprochene Wort analysieren muss, sondern auch Wortgrenzen finden muss, welche oftmals nicht eindeutig sind. Wie man anhand von Abbildung 1.3 und 1.4 sehen kann sind bei dem gesprochenen Satz „Der Spiegel spiegelt mich.“, die Wörter teilweise in sich verschwommen und erschweren dadurch das Erkennen jedes einzelnen Wortes immens. Noch hinzu kommen Faktoren wie Mehrdeutigkeiten der einzelnen Worte, die den ganzen Vorgang noch zusätzlich erschweren.

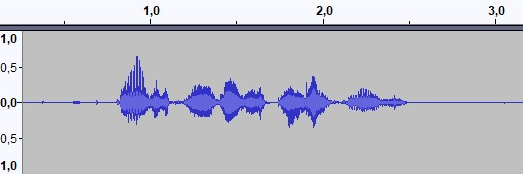
****

Abbildung 1.3

****

Abbildung 1.4

Ein weiterer Faktor, der die ganze Sache noch zusätzlich erschwert ist, das Mikrofon in welcher der Ton hineingesprochen wird, da nicht jedes Mikrofon gleich ist und der Sprecher mit dem gleichen Abstand bzw. den gleichen Umgebungsgeräuschen hineinspricht. In Abbildung 1.5 und 1.6 kann man sehen wie Umgebungsgeräusche einen ganzen Satz komplett verfälschen können. Beim Vergleich der beiden Aufnahmen kann man einen deutlichen Unterschied erkennen, vor allem das Grundrauschen bei Abbildung 1.6 macht es für den Algorithmus der Spracherkennung enorm schwierig. Man kann zwar noch einige Ähnlichkeiten erkennen, jedoch ob es sich bei den beiden Abbildungen um den gleichen Satz handelt ist für das menschliche Auge reine Spekulation.

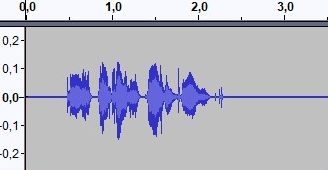


Abbildung 1.5 „Der Spiegel spiegelt mich.“

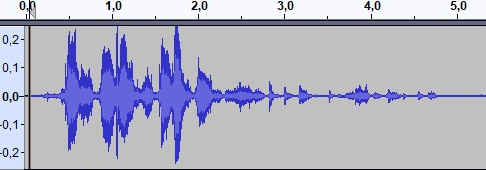


Abbildung 1.6„Der Spiegel spiegelt mich.“ + Störgeräuschen (Musik im Hintergrund)

**Mustererkennung**

Lösungsansätze für diese vielseitigen Probleme ist die Mustererkennung. Das Ziel der Mustererkennung ist es, aus der aufgenommenen Sprache Ähnlichkeiten, Wiederholungen und Regelmäßigkeiten zu erkennen. Man verwendet Mustererkennung jedoch nicht nur in der Spracherkennung, sondern auch in OCR Systemen, Überwachung, Biometrie, Gesichtserkennung und beim Fingerabdruck. Bei der syntaktischen Mustererkennung versucht man Dinge durch mehrmaliges Auftreten derselben Indikatoren zu beschreiben. Ein Beispiel zur näheren Erläuterung wäre hierfür die Unterscheidung zwischen Apfel und Birne. Der Apfel hat die Attribute rot und rund, die Birne hat die Attribute grün und oval. Durch diese Kategorisierung durch diese eindeutigen Attribute, kann man genau sagen um welches Wort es sich handelt. Jedoch kann man dadurch noch keine klare Zuordnung erzielen, daher muss man mit Wahrscheinlichkeiten versuchen eine Zuordnung zu finden. Eine weitere Methode stellt die statische Mustererkennung dar, bei dieser Methode versucht man nicht durch vordefinierte Regeln vorzugehen, sondern man versucht durch mithilfe von Zahlenwerten, welche in einen Merkmalsvektor zusammengefasst werden, die Wörter eindeutig zuzuweisen. Eine Kombination aus diesen beiden (syntaktische Mustererkennung und statischer Mustererkennung) ist die strukturelle Mustererkennung. Ein Beispiel, bei dem dieses Verfahren sehr oft angewendet wird, ist die Gesichtserkennung.

**Neuronale Netze**

Künstliche neuronale Netze werden ebenso immer bedeutender, da diese künstliche Intelligenz selbstständig neu dazulernen kann. Zum Beispiel das Entwickeln neuer Verbindungen oder das Löschen bestehender Verbindungen. Dieses neuronale Netzwerk lernt, indem es aufgrund einer vorgegebenen Regel handelt bzw. sich aufgrund dieser selbst verbessert. Es gibt drei verschiedene Arten des Lernen eines neuronalen Netzwerken.

Beim „Supervised Learning“, auch genannt Überwachtes Lernen wird jede Differenz von der Eingabe und der erwartenden Ausgabe angezeigt und dem Netz als neuen Baustein übergeben. Aufgrund dieser Differenz verändert das Netz seine anfangs festgelegten Regeln, kurz gesagt es lernt dazu. Die Eingaben werden einer fixen Kategorie übergeben. Man spricht von zwei verschiedenen Problemfällen, die hier auftreten können, zum einen Klassifikationsprobleme und zum anderen Regressionsprobleme. Bei Klassifikationsproblemen geht es darum, dass der Output nur wenige diskrete Werte annehmen kann zum Beispiel zu bestimmen ob eine E-Mail als Spam kategorisiert wird oder nicht und zu unterscheiden ob ein Objekt ein Flugzeug oder Schiff ist (Abbildung 1.5).

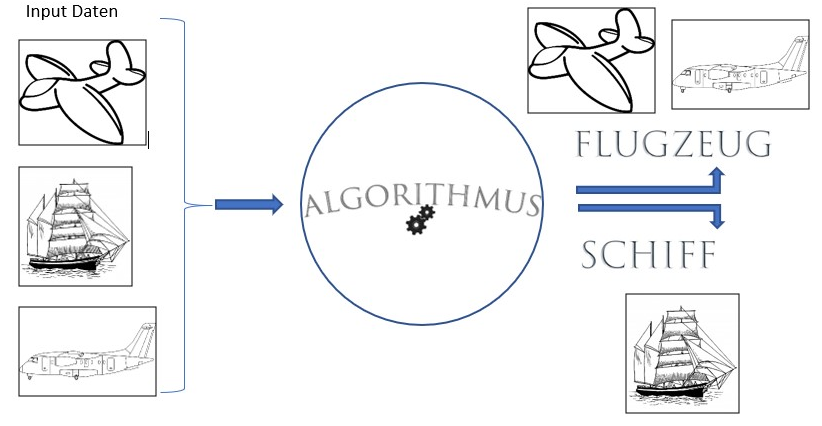


Abbildung 1.5

Bei Regressionsproblemen versucht man aufgrund von verschiedenen Eingangsvariablen einen gewissen Sachverhalt zu prognostizieren. In der Sprache könnte man aufgrund der eingefangenen Töne eine Regressionsanalyse durchführen, um ein Wort bzw. Satz zu erhalten.

Das Gegenstück vom „überwachten Lernen“ ist das „unüberwachte Lernen (Unsupervised Learning)“. Das Netz versucht, aufgrund der anfangs bestimmten Regeln sich selbst zu optimieren und mit den Gegebenheiten klar zu kommen. Das neuronale Netz findet sich durch Ähnlichkeiten der Eingangsdaten zurecht, welche durch verschiedene Eigenschaften bestimmt werden. Ein Algorithmus, der hier sehr oft verwendet wird, ist der „Clustering Algorithmus“. Es wird versucht gewisse Anomalien zu finden und diese richtig zuzuordnen. In dem Fall wird eine eigene Gruppe oder Zuordnungseinheit erstellt, anstatt von vordefinierten Gruppen. In Abbildung 1.2 kann man wieder ein Beispiel mit denselben Input Daten betrachten, wobei diesmal der Algorithmus selbst seine Zuordnungsgruppen erstellen muss in dem Fall Gruppe 1 für Flugzeuge und Gruppe 2 für Schiffe.

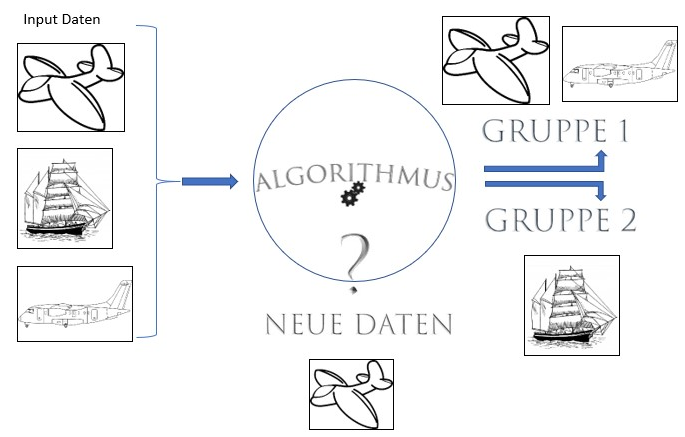
****

Abbildung 1.2

Beim „Reinforcement Learning“ (Abbildung 1.5) oder beim verstärkten Lernen werkt das Netz allein, jedoch wird dem Netz immer mitgeteilt, ob die Ausgabe korrekt bzw. inkorrekt war. Dem Netzwerk wird jedoch nicht mitgeteilt was genau falsch war. Das neuronale Netz versucht eine Strategie zu entwickeln, um seine korrekten Ausgaben zu maximieren. Es können verschiedenen Algorithmen zum Einsatz kommen, wie zum Beispiel der Monte-Carlo Algorithmus oder der „Temporal Difference Learning“ Algorithmus.

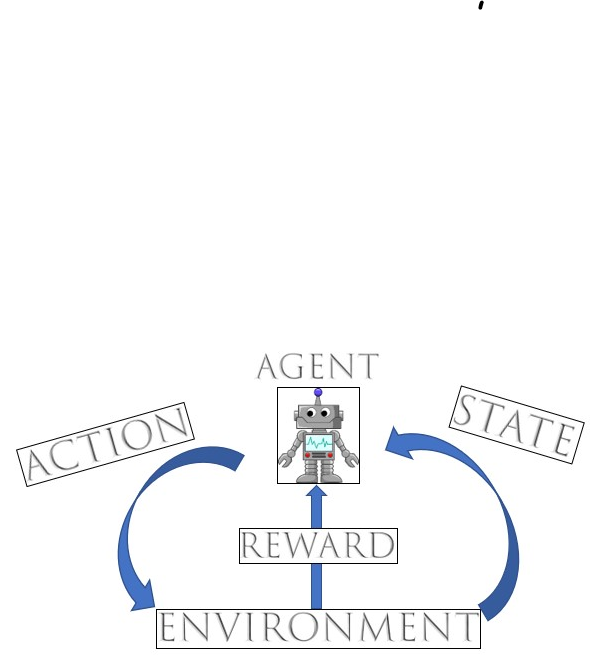
****

Abbildung 1.5

In diesem Szenario ist der Agent der, der alle Aktionen in einem bestimmten Simulationsszenario ausführt, zum Beispiel ein Spiel. Der State gibt die aktuelle Situation an, in der er sich befindet, da der Agent mehrere Aktionen ausführen kann, bevor der Zeitpunkt bestimmt wird, an dem er das Feedback bekommt, kann dies öfter durchlaufen werden. Zu einem gegebenen Zeitpunkt wird dem Agenten mitgeteilt, ob er etwas negativ oder positiv gemacht hat, was mit einer Belohnung bzw. Bestrafung gleichkommt. Hierdurch lernt der Agent, welche Aktionen sich gut auf das Geschehen ausgewirkt haben bzw. welche nicht.

**Algorithmen der Spracherkennung**

Es gibt verschiedene Algorithmen, die man zur Spracherkennung nutzt. Diese sollen die oben genannten Probleme so gut wie möglich beseitigen, um eine funktionierende Sprachsteuerung zu gewährleisten. Der erste Algorithmus ist der „Hidden Markov Model“ Algorithmus, welcher zur Spracherkennung am meisten genutzt wird.

Dieses Modell beruht auf dem Prinzip der Markov-Ketten. Für jeden einzelnen Laut (Phonem) wir ein sogenanntes HMM gebildet. Wenn man mehrere HMMs zusammensetzt kann man dadurch das wahrscheinlichste Wort bilden.

Um das Ganze ein wenig zu verdeutlichen ein kleines Beispiel in Abbildung 1.7. Hier haben wir als Beispiel das Wetter, welches ich aus Verständnisgründen in nur zwei Zustände gegliedert habe, nämlich sonnig und regnerisch. In der Abbildung 1.7, sieht man eine ganze Woche dargestellt von Montag bis Sonntag, mit dem jeweiligen vorherrschenden Wetter. Unter dieser Wochendarstellung findet man die Häufigkeit wie oft ein Ereignis aufgetreten ist, mit deren Wahrscheinlichkeit. In chronologischer Reihenfolge wurden die Ereignisse sonnig auf sonnig, sonnig auf regnerisch, regnerisch auf sonnig und regnerisch auf regnerisch, aufgelistet. Daraus ergibt sich dann ein Modell für die einzelnen Wahrscheinlichkeiten, welches man Word-HMM oder Markov Generation Model nennt (Abbildung 1.8)

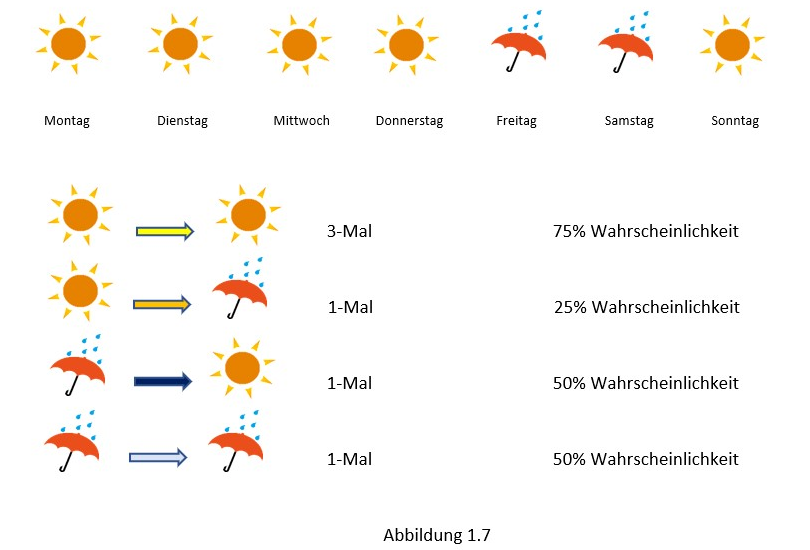


Abbildung 1.7

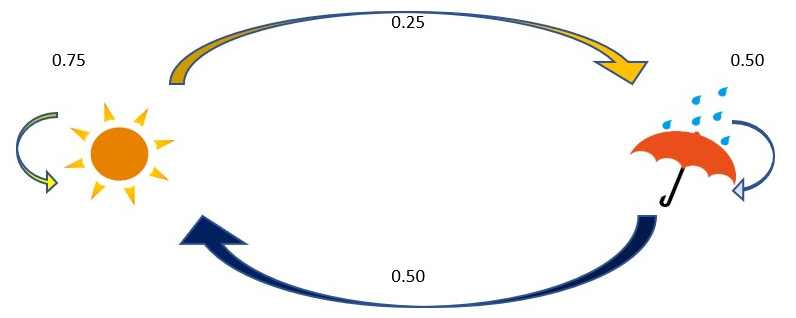


Abbildung 1.8

Das gleiche kann man bei der Sprache verwenden als weiteres Beispiel habe ich das Wort „laut“ genommen. Man verwendet das phonetische Alphabet, um die einzelnen Laute zu identifizieren, die deutsche Sprache hat ungefähr 40 Phoneme. Die meisten Sprachen haben um 20 – 60 Phoneme. Nachdem das eingehende Signal in seine Einzelteile der Phoneme bzw. in Frames zerteilt wurde, erhält man für jedes Einzelstück ein gewisses Phoneme, in unserem Beispiel „laut“ (Abbildung 1.9). Die Wahrscheinlichkeiten sind in dieser Abbildung nicht eingezeichnet, da die Wahrscheinlichkeit abhängig ist von dem Wortschatz und anderen Faktoren. Durch die Kombination von hochkomplexen mathematischen Formeln, kommt es zu einem Ergebnis des erkannten Wortes. Selbst wenn es von der Ideallösung leicht abweicht, aufgrund von Störsignalen oder Sprachvarietäten.

Abbildung 1.9

Ein weiterer Algorithmus wäre der Dynamic time warping (DTW) Algorithmus, dieser war in damaligen Zeiten sehr gut vertreten. Beim dynamischen time warping, geht es darum, Ähnlichkeiten zwischen zwei Sequenzen zu finden, wobei Zeit oder Geschwindigkeit variieren können. Für jedes Wort im Vokabular wird ein Referenzmuster abgespeichert und es wird immer paarweise die Distanz zwischen Referenzmuster und der Aufnahme ausgerechnet, man erkennt dies, wo diese Distanz am kleinsten ist. Für jedes Wort kann es mehrere Referenzmuster geben, jedoch bleibt der Vorgang derselbe, erfordert jedoch mehr Rechenleistung. Ein Beispiel für DWT wäre die Telefonauswahl bei älteren Telefonen, als man für einen Kontakt die Möglichkeit hatte diesen per Spracheingabe anzurufen. Zuerst wurde man aufgefordert den Namen ein paar Mal zu sagen, dass das Telefon ein Referenzmuster hat und dieses dann im Ernstfall mit dem gesprochenen Wort vergleichen kann. Bei mehreren Wörtern sogenannten Sequenzen mehrere Wörter verknüpft man mehrere der Einzelerkennungen. Dieser Algorithmus wurde jedoch durch den erfolgreichen „Hidden Markov Model“ Algorithmus verdrängt, da dieser eine bessere Genauigkeit für die Wörter bzw. Sätze aufweist, ebenso ist der HMM-Algorithmus besser auf mehrere Sprecher abgestimmt und muss nicht für jeden neuen Sprecher ein neues Referenzmuster erstellen.

**Implementierung der Sprachsteuerung in unser Projekt**

Wir haben uns dafür entschieden, dass die Sprachsteuerung die Tastatur zur Gänze ersetzt. Somit sind Einstellungen und Öffnen von Forms in unserem Projekt voll und ganz mit der Sprache möglich. Aufgrund einer testweisen Implementierung der Sprachsteuerung in das Spielgeschehen, um den Schläger nach rechts bzw. links zu bewegen haben wir abgesehen, da vom Eingang des Signals, der Analyse bis zur Kommandoausführung zu viel Zeit vergangen ist. Diese Zeitspanne wirkt sich nicht positiv auf das ganze Spiel aus, welches man eher als ein eher rasantes und schnelles Spiel verkörpern könnte. Um das bereits vorhandene Spracherkennungsmodul nutzen zu können, benutzen wir die von Microsoft bereitgestellte Assembly „System.Speech.dll“, welche als Verweis hinzugefügt wurde.

Die Sprachimplementierung ist für jede Form einzeln vorgesehen, welche auf differenzierte Art und Weise handelt basierend auf den Kommandos in deren jeweiligen Textdateien. Damit ist sichergestellt, dass bei einer aktiven, im Vordergrund laufenden Form nur die Befehle ausgeführt werden, die für diese Form zulässig sind und nicht beispielsweise andere Einstellungen getroffen werden in anderen Forms. Alle grammatikalischen Bausteine bzw. Sätze oder Wörter werden für jede Sprachmodulinstanz in einer Textdatei gespeichert. Als Beispiel hierfür wären die Kommandos für die Form des Client in Abbildung 1.0 und Abbildung 1.1.



Abbildung 1.0

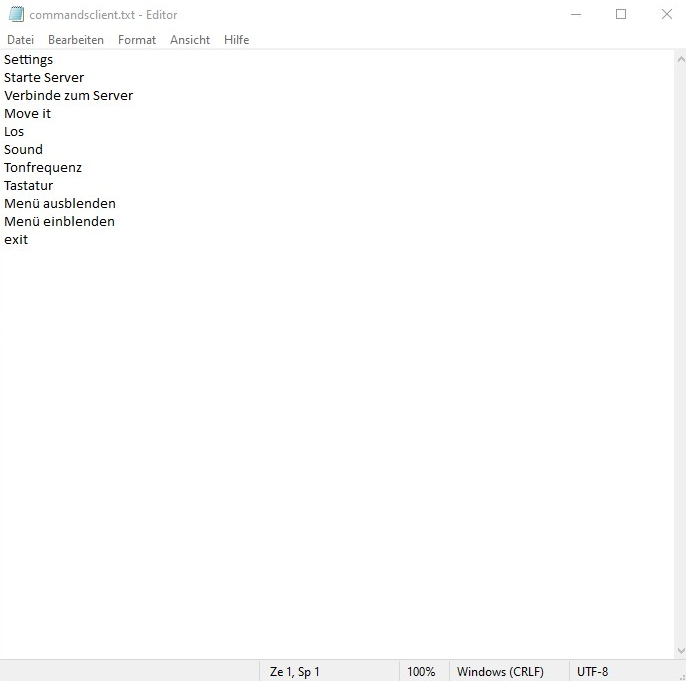


Abbildung 1.1

Bei der Initialisierung der Form wird das Spracherkennungsmodul (Abbildung 1.3) geladen, wobei es zuerst so konfiguriert wird, dass es Eingaben vom Standard-Audiogerät empfängt. Danach wird die Grammatik also die sprachlichen Bausteine geladen, welche bei Übereinstimmung von Eingangssignal mit der Textdatei einen Event erstellt, welcher für weitere Operationen zuständig ist. Zum Schluss dieser Initialisierung, wird noch festgelegt, dass das Modul fähig ist mehrere Befehle auszuführen.

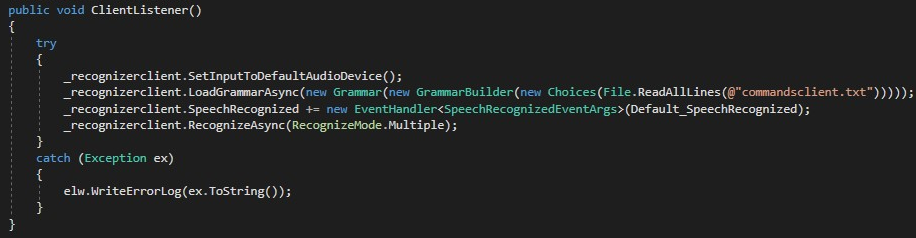


Abbildung 1.3

Im Falle, dass man das Fenster aus dem Fokus verliert und man andere Einstellungen bezüglich des Spiels oder generell am Computer vornimmt wurde die Sprachsteuerung so implementiert, dass sie bei Fokusverlust das Sprachmodul temporär gestoppt wird. Erst bei erneutem fokussieren der Form wird das Sprachmodul wieder auf aktiv gesetzt. In Abbildung 1.4 kann man sehen, dass das Modul wieder aktiv wird und in Abbildung 1.5 wird das Modul temporär gestoppt.

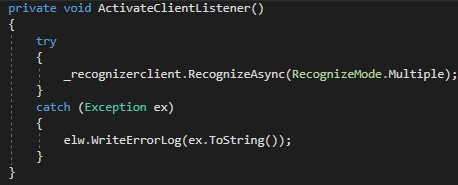


Abbildung 1.4

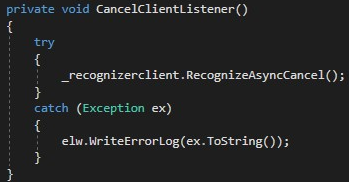


Abbildung 1.5

Das vorhin kurz erwähnte Event, wird dann ausgeführt, wenn ein grammatikalischer Baustein des festgelegten Wortschatzes in der Textdatei mit dem gesprochenen Wort übereinstimmt. In dieser Eventmethode führt man für das jeweilige Wort oder den Satz einen Codeabschnitt aus. In der Abbildung 1.4 und Abbildung 1.5, kann man sehen, dass durch „Starte Server“ der Server gestartet wird und im Zuge dessen, eine künstliche Erzeugung der menschlichen Stimme eine Antwort liefert. Da der Server in unserem Fall nur einmal gestartet werden darf, wird dieser Code beim zweiten Mal überspringen. Aus Gründen der Übersicht wurde dieser ausführbare Code immer in eigenen Methoden gegliedert

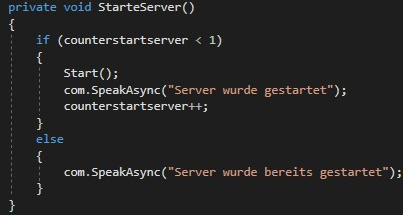


Abbildung 1.4

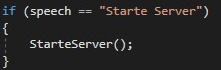


Abbildung 1.5

Innerhalb dieses Events wurde nochmals unterschieden, ob sich der Benutzer innerhalb eines „Tab“ befindet (Abbildung 1.6), um zu gewährleisten, dass man nicht unabsichtlich im Hintergrund Einstellungen trifft.

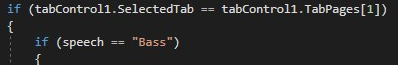


Abbildung 1.6