Karlsruhe Institute of Technology

Aufbau und Analyse eines Alice-Korpus

Bachelorarbeit
von

Sina Hampel

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Erstgutachter: Prof. Dr. Walter F. Tichy
Betreuender Mitarbeiter: Dipl.-Inform.Wirt Mathias Landhäußer

Bearbeitungszeit: 23. Juli 2012 – 08. November 2012

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Das Gesamtprojekt . . . 2
- 1.2. Ziel dieser Arbeit . . . 2
- 1.3. Aufbau der Arbeit . . . 2

2. Grundlagen **3**
- 2.1. Natürliche Sprache . . . 3
- 2.2. Textkorpora . . . 4
  - 2.2.1. Aufbau eines Korpus . . . 4
  - 2.2.2. Kategorisierung . . . 5
  - 2.2.3. Korpusannotation . . . 6
- 2.3. Alice . . . 6
- 2.4. Bestehende Textkorpora . . . 8

3. Versuchsaufbau und -durchführung **11**
- 3.1. Die Filmsequenzen . . . 11
  - 3.1.1. Beschreibung des Inhalts . . . 11
  - 3.1.2. Kriterien für die Filmauswahl . . . 13
  - 3.1.3. Erstellungsprozess . . . 14
- 3.2. Die Musterlösungen . . . 16
  - 3.2.1. Zweck . . . 16
  - 3.2.2. Linguistischer Aufbau . . . 16
  - 3.2.3. Gegenüberstellung von Musterlösung und Programmcode . . . 17
- 3.3. Aufgabenstellung . . . 19
- 3.4. Die Probanden . . . 20
- 3.5. Die Durchführung . . . 20

4. Vergleich und Analyse der Texte **23**
- 4.1. Linguistische Analyse . . . 23
- 4.2. Wort- und Satzanzahl . . . 24
  - 4.2.1. Aufteilung nach Programmierkenntnissen . . . 25
  - 4.2.2. Aufteilung nach Beaufsichtigung bei der Bearbeitung . . . 27
- 4.3. Markieren der Wortarten . . . 29

5. Zusammenfassung **35**

6. Ausblick **37**

Literaturverzeichnis **39**

Abbildungsverzeichnis **41**

Tabellenverzeichnis **43**

Anhang **45**
A. Fragebogen für Probanden 45
B. Alice-Skript für die Filmsequenz „Penguin“ 49
C. Alice-Skript für die Filmsequenz „Bunny“ 53
D. Musterlösungen 57
D.1. Musterlösung für die Filmsequenz ”Penguin” 57
D.2. Musterlösung für die Filmsequenz ”Bunny” 57
E. Beschreibungstexte 58
E.1. Text 1 58
E.2. Text 2 58
E.3. Text 3 59
E.4. Text 4 59
E.5. Text 5 59
E.6. Text 6 59
E.7. Text 7 60
E.8. Text 8 60
E.9. Text 9 60
E.10. Text 10 60
E.11. Text 11 61
E.12. Text 12 61
E.13. Text 13 61
E.14. Text 14 61

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 08.11.2012

.....

(**Sina Hampel**)

1. Einleitung

Während es in den Anfängen des Zeitalters der Informationstechnik besonders Informatiker waren, die sich mit Rechenmaschinen beschäftigten, besitzt heute eine große Zahl an Menschen aller Bevölkerungsschichten einen eigenen Computer. Auch Handys sind heute Computer – man denke an die Smartphones mit ihren vielfältigen Applikationen. Das Potential der Hardware kann dabei nur vollständig ausgeschöpft werden, wenn die Anwender in der Lage sind, zu programmieren. Informatiker können, im Gegensatz zu den meisten anderen Menschen, mit Hilfe von Programmiersprachen eigene Programme schreiben. Nicht-Informatiker, die gerne selbst ein Programm schreiben wollen, haben dagegen oft keine oder nur geringe Programmierkenntnisse, die nicht ausreichen, um ihre Ziele umzusetzen.

Ein Ansatz, diesen Menschen das Erstellen eines eigenen Programmes zu ermöglichen, ist die Entwicklung einer Programmierumgebung, die einen Text in natürlicher Sprache entgegennimmt und diesen in vom Computer interpretierbaren Code umwandelt. Dabei dient die natürliche Sprache selbst als Programmiersprache. Das ist naheliegend, da Menschen daran gewöhnt sind, Sachverhalte in natürlicher Sprache zu erklären. Das Ziel ist dabei, dass der Computer die Fähigkeit erhält, diese Erklärungen zu verstehen.

Dies klingt bei der ersten Betrachtung einfacher, als es jedoch ist. Die Schwierigkeit hierbei ist die Vielfalt natürlicher Sprache, die für einen Menschen leicht zu bewältigen ist, für den Computer jedoch eine große Herausforderung darstellt. Ein Satz kann mehrere Bedeutungen haben, von welchen der Mensch die richtige meist sofort aus dem Kontext heraus erkennen kann. Der Mensch merkt dabei nicht bewusst, welche Schritte in seinem Kopf ablaufen, die Interpretation wird als atomar wahrgenommen. Auf dem Computer hingegen muss dieser Denkprozess soweit es möglich ist emuliert werden, was einen sehr großen Aufwand darstellt. Ein Beispiel für ein Projekt, eine natürlichsprachliche Programmiersprache zu entwickeln, ist Pegasus [KM06] von der Technischen Universität Darmstadt.

Bei einer abgeschlossenen Entwicklung eines natürlichsprachlichen Programmiersystems würde dieses unter anderem die wesentlich einfachere Formulierung der Sachverhalte ermöglichen und eine größere Zielgruppe ansprechen. Außerdem würde sich die Redundanz deutlich verringern, denn bestehende Algorithmen könnten künftig in natürlicher Sprache formuliert werden und müssten nicht in verschiedenen Programmiersprachen erneut programmiert werden. Außerdem wäre die Dokumentation des Codes in weiten Teilen überflüssig, da dieser leichter zu verstehen wäre [KM06].

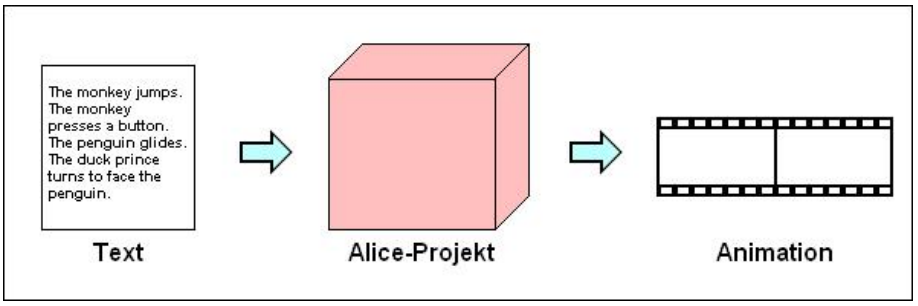

Abbildung 1.1.: Die Entstehung einer Animation

1.1. Das Gesamtprojekt

An diesem Institut soll ein natürlichsprachliches Programmiersystem entwickelt werden, das mit dem Programm Alice arbeitet. Alice ist eine Programmierumgebung und objektorientierte Programmiersprache, mit der Animationen aus 3D-Objekten erstellt werden können [CAB+00]. Das Programm wird in Kapitel 2.3 näher vorgestellt.

Das zu entwickelnde System soll natürliche Sprache entgegennehmen, diese in Alice-Quellcode umwandeln, sodass Alice daraus schließlich eine Animation, auch Alice-Welt genannt, erstellen kann. Der Prozess ist in Abbildung 1.1 dargestellt. Alice wird in diesem Falle verwendet, da es sich um eine objektorientierte Programmiersprache mit grafischer Darstellung der Objekte und Methoden handelt.

1.2. Ziel dieser Arbeit

Der erste Schritt in diesem Entwicklungsprozess ist das Sammeln von Texten in natürlicher Sprache aus einer gemeinsamen Domäne, die von unterschiedlichen Menschen verfasst wurden, um diese auf Gemeinsamkeiten und Unterschiede zu untersuchen. Dies ist notwendig, um eine Vorstellung davon zu erhalten, wie Menschen einen solchen Text formulieren würden. Ohne diesen Ansatz würde die Implementierung des Systems erschwert. Für dieses Ziel sollten die gesammelten Texte aus einer für unser Problem geeigneten Domäne stammen. Hier wird die schriftliche Beschreibung einer Filmsequenz verwendet, die mit Hilfe von Alice erstellt wurde. Die Verwendung von Alice ist aufgrund der Einbindung in das Gesamtsystem gegeben.

Eine solche Sammlung von Texten aus einer Domäne nennt sich Textkorpus, ein Begriff, der in Kapitel 2.2 definiert wird. Mit der Erstellung eines solchen Korpus und den dafür benötigten Vorbereitungen befasst sich diese Arbeit. Das entstandene Textkorpus bildet die Basis für das weitere Vorgehen bei der Entwicklung des Systems. Er dient den Entwicklern als Beispiel für die Verwendung natürlicher Sprache. Damit definiert er indirekt, wie später die Verbindung zwischen der natürlichen Sprache und dem Quellcode hergestellt wird, ohne genaue Entwurfs- oder Implementierungsdetails festzulegen. Somit kann das Korpus als ein Teil der Anforderungserhebung angesehen werden.

1.3. Aufbau der Arbeit

In Kapitel 2 dieser Arbeit werden zunächst die Grundlagen erklärt. Dabei wird als erstes der Begriff der natürlichen Sprache definiert. Danach werden Textkorpora und deren Aufbau erklärt und Alice vorgestellt. Zuletzt wird noch auf verwandte Arbeiten verwiesen.

In Kapitel 3 werden der Versuchsaufbau und seine Durchführung dargestellt. In Kapitel 4 werden die Beschreibungstexte analysiert. Abschließend folgen Zusammenfassung und Ausblick.

2. Grundlagen

In diesem Kapitel wird zunächst erklärt, was der Begriff der natürlichen Sprache bedeutet. Danach wird der Begriff des Textkorpus definiert, der Prozess des Aufbaus eines Textkorpus erläutert und verschiedene Arten von Textkorpora vorgestellt. Anschließend soll noch erklärt werden, was es bedeutet, ein Textkorpus zu annotieren.

Danach wird das Programm Alice vorgestellt, mit dessen Hilfe die Filmsequenzen erstellt wurden, die während der Versuchsphase von den Probanden beschrieben wurden. Zuletzt werden bestehende Textkorpora vorgestellt, um zu klären, in welche Richtungen auf diesem Gebiet schon geforscht wurde.

2.1. Natürliche Sprache

Um zu erklären, was Programmieren in natürlicher Sprache bedeutet, muss zunächst der Begriff der natürlichen Sprache definiert werden. Eine natürliche Sprache ist eine von Menschen gesprochene Sprache, die sich historisch entwickelt hat [Lew85]. Plansprachen wie Esperanto sind keine natürlichen Sprachen, da sie sich nicht historisch entwickelt haben.

Formale Sprachen, bei denen nicht die Kommunikation, sondern der mathematische Aspekt im Vordergrund steht, müssen von natürlichen Sprachen unterschieden werden, da natürlichsprachliche Ausdrücke oftmals nicht eindeutig sind. Die Interpretation ist dabei abhängig vom Kontext. Mehrdeutigkeit zeigt sich dabei auf mehreren Ebenen[1]:

- Mehrdeutigkeit auf Wortebene: *Er sitzt auf der Bank.* Dieser Satz kann einerseits bedeuten, dass er auf einer Bank (dem Möbelstück) sitzt. Andererseits kann der Satz bedeuten, dass er in einer Bank (dem Kreditinstitut) auf einer beliebigen Sitzgelegenheit sitzt. Eine dritte Bedeutung wäre, dass er auf dem Dach des Kreditinstituts sitzt.
- Mehrdeutigkeit auf Wortbildungsebene: *Fischfrau.* Dieses Wort kann zwei Bedeutungen haben, nämlich dass es sich um eine Frau handelt, die Fische verkauft, oder um eine Frau, die halb Frau und halb Fisch ist.
- Mehrdeutigkeit auf Semantikebene: *Der Mann sieht das Mädchen mit dem Fernrohr.* Auch dieser Satz kann auf unterschiedliche Weise interpretiert werden. Entweder

[1] Diese Beispiele stammen aus der Vorlesung *Sprache, Logik, Ontologien* am Institut für deutsche Sprache und Literatur der Universität Dortmund, gehalten von Prof. Gabriele Kern-Isberner und Prof. Angelika Storrer im Sommersemester 2006.

sieht der Mann durch das Fernrohr und erblickt dabei ein Mädchen, oder der Mann sieht ein Mädchen, das ein Fernrohr in der Hand hält.

An diesen Beispielen sieht man, dass die natürlichen Sprachen viel flexibler sind als die formalen Sprachen. Sie nutzen die Interpretationsfähigkeit des Menschen. Dies ist im Alltag kein Problem. Will man jedoch mit einer Maschine kommunizieren, führt dies unweigerlich zu Problemen, da diese exakte Anweisungen benötigen. Daher ist es für einen Computer schwierig, Sätze in natürlicher Sprache korrekt zu interpretieren. Dies ist die Herausforderung bei der Entwicklung eines natürlichsprachlichen Programmiersystems.

Ein weiteres Problem sind die weniger strengen Regeln der natürlichen Sprache. Fehlt bei einer Aussage in einer formalen Sprache ein Zeichen, so wird diese Aussage in aller Regel nicht verstanden. Nehmen wir als Beispiel die mathematische Aussage $3 + 5 = 8$. Entfernt man aus dieser Aussage das Pluszeichen, erhält man $3 \_ 5 = 8$. Diese Aussage wird weder von den meisten Menschen, noch von einem Computer so verstanden, wie sie gemeint war. Die rechnende Suchmaschine Wolfram Alpha [Wol] behandelt diese Eingabe beispielsweise wie die mathematische Aussage $3 * 5 = 8$, was nicht nur eine falsche Interpretation, sondern auch eine mathematisch falsche Aussage darstellt. Bei natürlicher Sprache ist dies anders: Der Satz „Dein Vater er ist“ [KL83] ist zwar kein korrektes Deutsch, wird jedoch von einem deutschsprechenden Menschen in der Regel verstanden. Auch diese Tatsache stellt bei der Entwicklung des Systems eine große Schwierigkeit dar, da aus der unscharfen, mehrdeutigen Formulierung eine exakte Formulierung generiert werden muss, ohne jedoch ihren Sinn zu verändern. Um dies zu erreichen, gibt es verschiedene Ansätze. Der erste ist, dass den Menschen, die das System bedienen, gewisse Einschränkungen vorgegeben werden müssen, mit denen die Sätze formuliert werden dürfen. Diesen Ansatz verfolgt die Universität Zürich bei ihrem Projekt Attempto Controlled English (ACE) [Kuh10]. Ein anderer Ansatz würde in einem größeren Programmieraufwand bestehen, um die Intention des Sprechers bei einer mehrdeutigen Äußerung zu bestimmen.

2.2. Textkorpora

Ein Textkorpus, oder kurz Korpus, ist eine Sammlung von natürlichsprachlichen Artefakten, die sich aus verschiedenartigem Material zusammensetzen kann. Das können aufgezeichnete Konversationen sein, Radiosendungen, schriftliche Veröffentlichungen, oder auch von Kindern geschriebene Texte [McE03]. McEnery definiert weiter, dass ein Korpus eine gut organisierte Sammlung darstellen sollte, mit Texten, die einem bestimmten *Sampling Frame* entsprechen, einem Rahmen, der festlegt, welche Daten sich zur Erfoschung bestimmter linguistischer Merkmale eignen. Da, vorausgesetzt es handelt sich um eine lebendige Sprache, niemals alle existierenden sprachlichen Artefakte gesammelt werden können, sollte der Korpus außerdem innerhalb seines Sampling Frames ausgewogen und repräsentativ sein. Da Korpora als Forschungsgrundlage dienen, sind sie meist maschinenlesbar, da die Verarbeitung einer Text- oder Audioquelle, die mehrere Millionen Wörter umfasst, unpraktikabel wäre.

2.2.1. Aufbau eines Korpus

Der erste Schritt beim Aufbau eines Korpus besteht darin, die Texte, aus denen es bestehen soll, zu sammeln. Hierbei muss man zwei Dinge entscheiden, nämlich aus welchen Themengebieten die Texte stammen sollen und welche Art von Sprache man untersuchen will. Diese Entscheidung fällt je nach zu erforschendem Gebiet unterschiedlich aus. Will man beispielsweise gesprochene Sprache untersuchen, ist es sinnlos, ein Korpus aus Zeitungsartikeln aufzubauen. Will man eine bestimmte Fachsprache untersuchen, sollte man Veröffentlichungen zu diesem Thema sammeln. Diese Eigenschaft des Korpusaufbaus

macht die Studie mit den Probanden, die in Kapitel 3 beschrieben wird, notwendig. Das liegt daran, dass es keine Texte gibt, die eine mit Alice erstellte Filmsequenz beschreiben. Da aber gerade mehrere solcher Texte benötigt werden, welche die gleiche Filmsequenz beschreiben und von verschiedenen Personen verfasst sind, müssen diese Texte zuerst angefertigt werden, bevor das Korpus aufgebaut werden kann.

Wenn die gewünschten Texte vorliegen, werden gewöhnlich noch einige Arbeitsschritte durchgeführt, bis aus diesen Texten ein Korpus entstanden ist. Im nächsten Schritt müssen die gesammelten Texte vereinheitlicht werden. Das bezieht sich einerseits auf die Dokumentstruktur, andererseits auch auf möglicherweise vorhandene Metadaten. Dokumentstruktur bedeutet beispielsweise, dass je nach Text die Formatierung der Schrift unterschiedlich sein kann (zum Beispiel fett, kursiv, verschiedene Schriftarten). Bei Metadaten kann es sich zum Beispiel um Angaben zum Autor, Datum, oder Seitenumbrüche handeln [Bub11].

Sind die Texte vereinheitlicht, müssen sie im nächsten Arbeitsschritt segmentiert werden. Das bedeutet, dass sie in kleinere Einheiten aufgespaltet („tokenisiert“) werden. Das Token ist dabei als kleinste Einheit, die durch Leerzeichen oder Interpunktion begrenzt wird, definiert [Bub11]. Nach der Segmentierung werden die Token wieder zu Sätzen zusammengefasst. Dabei ist es wichtig, Satzgrenzen zu erkennen und von anderen Punktzeichen zu unterscheiden. Andere Fälle, in denen ein Punkt am Ende eines Wortes steht, können eine Kardinalzahl oder eine Abkürzung sein. In einem solchen Fall ist der Satz jedoch nicht zu Ende, was durch verschiedene Verfahren vom Computer erkannt werden kann.

2.2.2. Kategorisierung

Korpora können aufgrund verschiedener Merkmale kategorisiert werden. Dabei unterscheidet McEnery [McE03] die folgenden Arten von Korpora:

- Monolinguale Korpora enthalten Texte in einer einzigen Sprache.
- Vergleichbare Korpora enthalten mehrere monolinguale Korpora unterschiedlicher Sprachen, die den gleichen Sampling Frame verwenden, um die enthaltenen Sprachen vergleichen zu können. Dabei sind die Texte in den verschiedenen Sprachen nicht identisch.
- Parallele Korpora enthalten Texte in einer einzigen Sprache, die dann in andere Sprachen übersetzt werden. Die Texte sind hierbei in allen Sprachen identisch.

Außerdem unterscheidet McEnery zwischen Korpora, die aus Schriftsprache, gesprochener Sprache, oder einer Mischung aus beiden bestehen. Dabei kann die gesprochene Sprache ausschließlich in Form von Audiodateien vorliegen, oder, falls die Audiodateien nicht (mehr) existieren, nur als Transkript. Weitere Korpusarten sind:

- Multimodale Korpora sind Sprachkorpora, die zusätzlich mit Prosodien[2], Mimik und Gestik angereichert sind.
- Baumbanken (Treebanks) sind schriftliche Korpora, die mit Syntaxbäumen annotiert sind. Das bedeutet, dass die syntaktische Struktur eines Satzes hierarchisch dargestellt ist [CEE+01, LZ06]. Ein Beispiel für eine solche Baumbank ist das Penn Treebank Project [MSM93], die unter anderem das Brown Korpus, das in Kapitel 2.4 vorgestellt wird, und Texte aus dem Wall Street Journal in annotierter Form enthält.

[2]Prosodien sind für die Gliederung der Rede bedeutsame sprachlich-artikulatorische Erscheinungen wie Akzent, Intonation, Pausen, oder Ähnliches [dud].

Etikett	Wortart
NN	Substantiv Singular
VBP	Verb Präsens (nicht 3. Person Singular)
VBD	Verb Vergangenheit
CC	Konjunktion
IN	Präposition
DT	Determinierer
.	Punkt

Tabelle 2.1.: Etiketten und damit bezeichnete Wortarten des University of Illinois Wortart-Markierers [ill]

Außerdem unterscheidet man statische und Monitorkorpora. Statische Korpora sind abgeschlossen und werden nicht mehr erweitert. Monitorkorpora werden ständig mit neuen Texten ergänzt [Teu98]. Die Art des verwendeten Korpus sollte sich dabei nach dem erforschten Gebiet richten.

2.2.3. Korpusannotation

Ein annotiertes Korpus ist ein Textkorpus, das mit verschiedenen Arten linguistischer Informationen angereichert ist [McE03]. Das bedeutet, dass beispielsweise für jedes Wort die zugehörige Wortart, sowie der Kasus, Genus und Numerus mit angegeben werden. Der Fachbegriff hierfür ist Wortart-Markierung (Part-Of-Speech Tagging), wofür mehrere Werkzeuge zur automatischen Annotation zur Verfügung stehen. Ein Beispiel für ein solches Werkzeug ist der Stanford Log-linear Part-Of-Speech Tagger [TKMS03]. Die Annotation dient dazu, das Korpus besser erforschen zu können.

McEnery nennt als Vorteile der Annotation die schon genannte einfachere und schnellere Verarbeitung der Daten, außerdem Wiederverwendbarkeit, Multifunktionalität und explizite Analysen.

Die Universität von Illinois in Urbana-Champaign bietet ebenfalls einen Wortart-Markierer an [ill], der an dem Satz *Men have landed and walked on the moon.* die folgende Annotation vornimmt:

> NN/Men VBP/have VBD/landed CC/and VBD/walked IN/on DT/the NN/moon ./.

Dabei zeigt Tabelle 2.1, welche Etiketten welche Wortarten bezeichnen.

An diesem Beispielsatz kann man sehen, dass die Wortarten dieses englischen Satzes vom Computer fehlerfrei erkannt werden. Insgesamt existieren bei dem Wortart-Markierer der Universität von Illinois 36 Etiketten für die verschiedenen Wortarten und 11 Etiketten für verschiedene Satz- und Sonderzeichen.

2.3. Alice

Bei dem Programm Alice handelt es sich um eine interaktive 3D-Programmierumgebung und Programmiersprache, die es dem Benutzer erlaubt, mithilfe von vorgefertigten 3D-Objekten eine Animation zu erstellen. Alice wird seit 1999 von der Carnegie Mellon University entwickelt. Die Animationen können interaktiv gestaltet oder als Video exportiert werden. Alice wurde dazu entwickelt, Menschen eine Einführung in die objektorientierte Programmierung zu geben. Dabei sollen vor allem grundlegende Konzepte gelehrt werden [CAB+00].

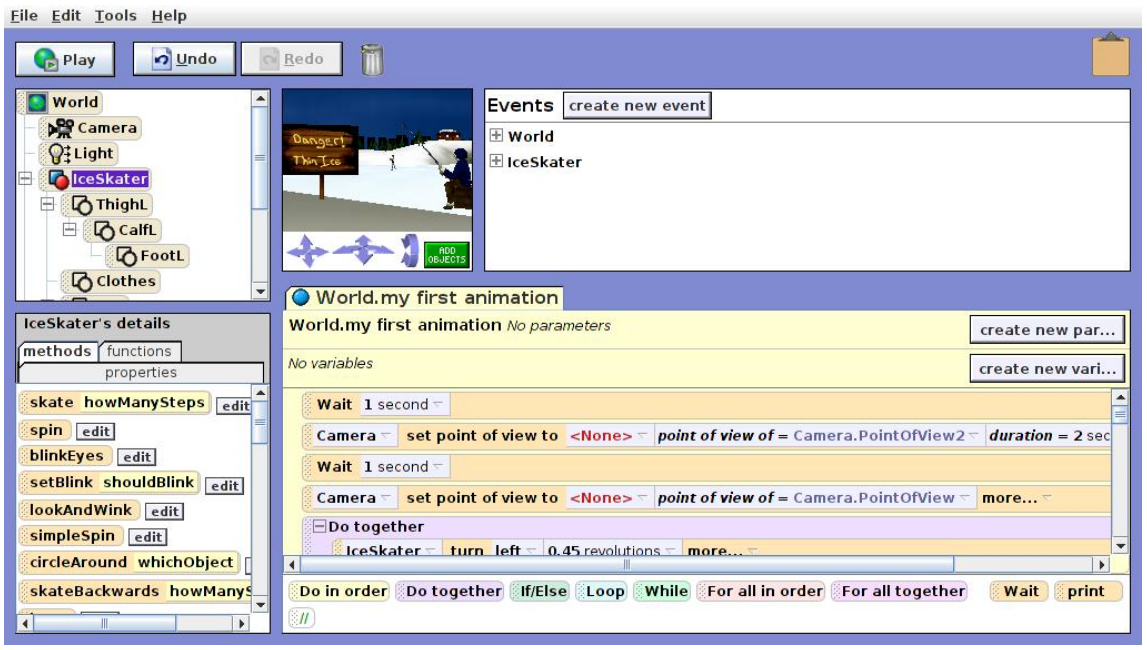

Abbildung 2.1.: Bildschirmfoto aus Alice 2.0

In Abbildung 2.1 sieht man die Benutzeroberfläche von Alice, wie sie sich darstellt, wenn eine Alice-Welt geöffnet ist und bearbeitet wird. Wie man erkennen kann, ist die Oberfläche in fünf Hauptbestandteile aufgeteilt. Beginnt man links oben und folgt man ihnen im Uhrzeigersinn, so sind das:

- Das Objektfenster: hier werden die in der Welt platzierten Objekte angezeigt. Sie sind zusammen mit ihren Unterobjekten als Baumstruktur angeordnet. Hier ist das Objekt „IceSkater“ ausgewählt.
- Die Welt: sie ist statisch dargestellt, die Animation lässt sich jedoch über den Knopf mit der Aufschrift „Play“ (am oberen Bildschirmrand) starten.
- Das Ereignisfenster: durch Mausklicks oder durch Starten der Welt können verschiedene Ereignisse auf den Objekten ausgelöst werden, wie etwa das Ausführen einer Methode oder eine Zustandsänderung.
- Das Methodenbearbeitungsfenster: in diesem Fenster lassen sich die Methoden bearbeiten. Es wird ein Pseudocode angezeigt, der durch Drag and Drop erweitert werden kann. Parameter lassen sich mithilfe von Dropdown-Listen bearbeiten.
- Die Objektdetails: in diesem Fenster werden die Methoden und Eigenschaften des im Objektfenster ausgewählten Objektes angezeigt. Diese lassen sich dann in das Methodenbearbeitungsfenster ziehen.

Im Programm lassen sich die 3D-Objekte zunächst in eine bestehende Welt setzen, die einen Boden als Bezugssystem besitzt. Dabei lässt sich die Größe, Ausrichtung und Position der Objekte frei festlegen. Sind die Objekte platziert, können sie in einem nächsten Schritt animiert werden. Das bedeutet, dass der Welt, wie die zu speichernde Datei bezeichnet wird, Methoden hinzugefügt werden, die teilweise vorgefertigt sind. Der Benutzer kann auch eigene Methoden definieren, die sich aus den bestehenden Methoden zusammensetzen lassen. Es existieren außerdem grundlegende Konzepte wie Schleifen oder `If`- und `Else`-Sprünge.

Die bereits bestehenden Methoden sind zum größten Teil bei allen Objekten gleich. Die wichtigsten dieser Methoden lassen das Objekt sich in eine Richtung bewegen, sich um sich

Korpus	**Brown**[FK79]	**LOB**[JAGL86]	**BNC**[Bur07]	**ICE**[ice]
Sprache	amerikanisches Englisch	britisches Englisch	britisches Englisch	verschieden
Medium	schriftlich	schriftlich	schriftlich und mündlich	schriftlich und mündlich
Beginn	1963	1970	1991	1990
Umfang	1 Mio. Wörter	1 Mio. Wörter	100 Mio. Wörter	je 1 Mio. Wörter
Verwendung	Lexikographie, Syntaxforschung, Statistik, Wortschatzforschung	Lexikographie, Syntaxforschung, Statistik, Wortschatzforschung	Lexikographie, Statistik, Spracherkennung	Vergleiche

Tabelle 2.2.: Kerneigenschaften der vorgestellten Korpora

selbst oder um eine Achse drehen. Auch verschiedene Teile des Objekts, wie zum Beispiel Körperteile eines Tieres, lassen sich bewegen, um so komplexere Bewegungen erstellen zu können. Bei einigen Objekten lässt sich zusätzlich die Farbe ändern, um so zum Beispiel den Effekt einer eingeschalteten Lampe (die Farbe wird von schwarz auf gelb geändert) simulieren zu können. Dieser Aufbau lässt sich in Kapitel 2.1 der Bachelorarbeit von Oleg Peters genauer nachlesen [Pet12].

Die Entwicklung des Gesamtsystems basiert auf Alice, und da zunächst ein Alice-Korpus benötigt wird, kam Alice auch bei dieser Arbeit zur Anwendung. Alice wurde im Gesamtprojekt verwendet, da der Quellcode natürlichsprachlichem Englisch sehr ähnlich ist, während sich andere Programmiersprachen sehr von natürlicher Sprache unterscheiden. Ein weiterer Grund ist, dass die Objekte und Methoden der realen Welt entnommen sind. Die Handlung kann also von den Benutzern ohne die Kenntnis eines speziellen Vokabulars beschrieben werden.

Im Verlauf der Arbeit hat sich gezeigt, dass sich mit Hilfe von Alice auch ein Textkorpus leicht erstellen lässt. Das liegt daran, dass man sehr einfach eine Musterlösung anfertigen kann, da, wie bereits zuvor erwähnt, das Skript englischem Text sehr ähnlich sieht. Außerdem sind die Texte leicht auf Vollständigkeit zu überprüfen.

2.4. Bestehende Textkorpora

Da sich, wie bereits erwähnt, ein Textkorpus zur Analyse von natürlicher Sprache gut eignet, sind bei den verwandten Arbeiten bestehende Textkorpora von Bedeutung. Auch diese Arbeit beschäftigt sich mit dem Aufbau eines Textkorpus und seiner Analyse.

Es gibt eine Reihe von bekannten Textkorpora, die teilweise bereits vor fünfzig Jahren aufgebaut wurden. Hier sollen das Brown Corpus, das British National Corpus (BNC), das LOB-Corpus und das International Corpus of English (ICE) betrachtet werden. Diese Korpora bestehen alle aus englischen Texten, eine Eigenschaft, die sie mit dem Alice-Korpus, welches in dieser Arbeit aufgebaut wurde, gemeinsam haben. In Tabelle 2.2 sind die Kerneigenschaften der genannten Korpora übersichtlich dargestellt.

Zunächst betrachten wir das Brown Corpus. Es handelt sich dabei um ein Korpus mit Texten aus englischer Prosa, die im Laufe des Jahres 1961 in den USA veröffentlicht wurden. Bei den Verfassern handelte es sich um Muttersprachler des Amerikanischen Englisch. Das Korpus besteht aus ungefähr einer Million Wörter. Das Korpus ist in 500 Schriftstücke mit

jeweils 2000 Wörter aufgeteilt. Das Korpus beschränkt sich dabei nicht auf ein bestimmtes Thema, sondern umfasst eine große Vielfalt an prosaischen Texten aus verschiedenen Themengebieten. Die Texte wurden während der 1960er Jahre an der Brown University in Providence, Rhode Island, gesammelt und zu einem Korpus zusammengefasst. In den folgenden Jahren wurden bei den Texten nachträglich die Wortarten markiert [FK79].

Das Lancaster-Oslo-Bergen-Corpus (LOB-Corpus) wurde von den Universitäten in Lancaster, Oslo und Bergen in den 1970er Jahren als gemeinsames Projekt aufgebaut. Da es als britisches Gegenstück zum Brown Corpus konzipiert ist, lassen sich einige Parallelen beobachten. Das LOB-Corpus beinhaltet, genau wie das Brown Corpus, Texte aus verschiedenen Themengebieten und umfasst ungefähr eine Million Wörter. Auch dieses Korpus besteht aus 500 Schriftstücken, die jeweils 2000 Wörter umfassen. Der Unterschied zum Brown Corpus besteht darin, dass das LOB-Corpus ausschließlich aus Texten in britischem Englisch besteht. Auch das beim LOB-Corpus wurden die Wortarten markiert [JAGL86].

Das British National Corpus (BNC) ist im Gegensatz zu den beiden bereits vorgestellten Korpora sehr viel umfangreicher. Es entstand in den 1990er Jahren als Zusammenarbeit der Oxford University Press, der Longman Group Ltd, Chambers Harrap, der Universitäten von Oxford und Lancaster, sowie der British Library. Das BNC wurde für eine Vielzahl von Anwendungsbereichen konzipiert, unter anderem auch für die automatische Verarbeitung natürlicher Sprache (*Natural Language Processing*, NLP). Auch das BNC enthält Texte aus verschiedenen Themengebieten, umfasst dabei aber ungefähr 100 Millionen Wörter. Die Texte stammen aus Zeitungen, Zeitschriften und wissenschaftlichen Publikationen. Das BNC unterscheidet sich von den anderen vorgestellten Korpora auch dahingehend, dass es sich bei etwa 10% der Inhalte um aufgezeichnete Konversationen handelt. Diese liegen als Transkriptionen vor. Die Wortarten wurden auch beim BNC markiert [Bur07].

Das International Corpus of English (ICE) unterscheidet sich stark von den anderen vorgestellten Korpora. Es umfasst nicht nur eine Variante der englischen Sprache, sondern soll verschiedene Varianten einander gegenüberstellen. Derzeit gibt es 12 eigenständige ICE-Korpora, die jeweils eine regionale Variante der englischen Sprache abdecken. Jedes Korpus umfasst eine Million Wörter und umfasst sowohl mündliche als auch schriftliche Artefakte. Das ICE wird seit 1990 als Zusammenarbeit verschiedener Arbeitsgruppen aufgebaut, die sich an verschiedenen Standorten mit der dort vorherrschenden Variante der englischen Sprache beschäftigen. Dabei steht der Aspekt des Vergleichens der Varianten im Vordergrund [ice].

Es gibt auch Textkorpora, die eigens dafür aufgebaut wurden, um als Trainingsdaten für NLP-Anwendungen zu dienen. Dieses Gebiet stellt einen wichtigen Bestandteil unseres Systems dar. Soll die natürliche Sprache vom Computer in Quellcode übersetzt werden, muss sie zuerst verarbeitet werden. Ein Beispiel für ein solches Textkorpus ist das Web Text Corpus, das an der Universität von Sydney aufgebaut wurde [LC06]. Hierfür wurden englische Webseiten verschiedener Themengebiete heruntergeladen. Das Korpus umfasst etwa 10 Milliarden Worte. Während des Vergleichs des Web Text Corpus mit anderen Methoden, wie etwa auf Texte im Web mit Suchmaschinen zuzugreifen, oder die Verwendung eines Korpus aus Printmedien, fand man heraus, dass das Web Text Corpus auf einigen Gebieten bessere Ergebnisse erzielt. Ein Beispiel dafür ist die kontextsensitive Verbesserung von Rechtschreibfehlern. Auf dem Gebiet der Synonymfindung erzielt das Web Text Corpus gleich gute Ergebnisse wie die anderen Methoden.

Die vorgestellten Korpora dienen entweder der allgemeinen Erforschung der englischen Sprache (Brown Corpus, LOB-Corpus, ICE) oder auch der Forschung auf dem Gebiet der Spracherkennung (BNC, Web Text Corpus). Auf diesem Gebiet soll zwar auch das in dieser

Arbeit aufgebaute Alice-Korpus Anwendung finden, jedoch wird in diesem Fall ein spezielles Teilgebiet der Spracherkennung betrachtet. Dabei handelt es sich um die Beschreibung einer Alice-Animation, die in Alice-Quellcode übersetzt werden soll. Daher wurden für den Aufbau des Alice-Korpus Beschreibungstexte zu Alice-Animationen benötigt. Es ist also nicht möglich, auf diesem Gebiet mit einem bestehenden Korpus zu forschen.

3. Versuchsaufbau und -durchführung

Um das Ziel dieser Arbeit, den Aufbau eines Alice-Textkorpus, zu erreichen, wurde eine Studie mit mehreren Testpersonen durchgeführt. Die Grundidee dieser Studie bestand darin, dass Testpersonen eine Filmsequenz in natürlicher Sprache schriftlich beschreiben sollten. Diese Filmsequenz wurde eigens für die Studie erstellt.

Zunächst wurde als Versuchsvorbereitung das Programm Alice verwendet, um die angesprochene Filmsequenz zu erstellen. Danach wurde eine zweite Filmsequenz angefertigt, um den Testpersonen, die möglicherweise keine Vorstellung davon haben, wie eine solche Beschreibung anzufertigen ist, eine Hilfestellung zu geben. Hierzu wurde den Testpersonen ebenfalls eine zuvor angefertigte Musterlösung vorgelegt, welche diese Filmsequenz so genau wie möglich beschreibt.

Für die Durchführung wurde ein Fragebogen erstellt, der von den Testpersonen ausgefüllt werden und ihnen gleichzeitig als genaue Anleitung für ihr Vorgehen dienen sollte. Einen großen Teil dieses Fragebogens nahm dabei die Beschreibung der Filmsequenz durch den Probanden ein. Der vollständige Fragebogen befindet sich in Anhang A.

In diesem Kapitel werden die Filmsequenzen mit ihrem Inhalt und und ihrem Erstellungsprozess erklärt, danach werden die Musterlösungen zu diesen Filmsequenzen vorgestellt, und schließlich die Aufgabenstellung, die Probanden und die Durchführung genauer beschrieben.

3.1. Die Filmsequenzen

Für die Erstellung der Filmsequenzen wurde Alice verwendet. Dies war notwendig, weil das zu entwickelnde System auf Alice aufbaut. Die Erstellung einer Filmsequenz geschieht einfach per Drag-and-Drop und das Ergebnis kann sofort angeschaut werden, was sich bei einer Filmsequenz, die für die Testpersonen später möglichst anschaulich sein soll, als Vorteil erweist. Trotz dieser simplen Art der Programmierung ist der so entstandene Code einsehbar, was sowohl für die Erstellung der Musterlösung als auch für weiterführende Arbeiten, bei denen die entstandenen Beschreibungen mit dem Programmcode verknüpft werden sollen, erforderlich ist.

3.1.1. Beschreibung des Inhalts

Zuerst soll die Welt „Penguin“ betrachtet werden. Wie man in Abbildung 3.1 sieht, befinden sich in der Welt drei Tierfiguren, ein Pinguin, ein Affe und eine Ente, welche die

Abbildung 3.1.: Aufbau der Welt „Penguin“

Abbildung 3.2.: Aufbau der Welt „Bunny“

drei Akteure darstellen. Alle Aktionen im Video gehen ausschließlich von diesen drei Akteuren aus. Weitere Objekte sind eine Fernbedienung, eine Glühbirne, ein Eimer und eine Sonnenblume. Während die Sonnenblume und die Fernbedienung unbewegte Objekte darstellen, die an keiner Aktion beteiligt sind, wird der Zustand der restlichen zwei Objekte zumindest von den Akteuren bei deren Aktionen verändert.

Wenn die Filmsequenz beginnt, dreht sich die Ente in Richtung des Affen, dem sie einen Befehl gibt, woraufhin dieser seufzt und die Glühbirne einschaltet. Danach dreht sich der Pinguin in Richtung des Eimers, legt sich auf den Bauch und rutscht den Boden entlang, wobei er den Eimer umwirft. Daraufhin dreht sich die Ente in Richtung des Pinguins und tadelt diesen, indem sie ihr Zepter schwingt. Daraufhin lacht der Affe und schaltet die Glühbirne wieder aus.

Der Aufbau der Welt „Bunny“, den man in Abbildung 3.2 sieht, unterscheidet sich bei genauer Betrachtung nur oberflächlich von dem der ersten Welt. Es befinden sich wieder Tierfiguren als Akteure in der Welt, diesmal jedoch nur zwei, ein Kaninchen und ein Frosch. Das einzige unbewegte Objekt ist in dieser Welt die Palme. Zwei weitere Objekte, deren Zustand sich zwar ändert, die jedoch selbst keine Aktionen ausführen, sind der Briefkasten und der Brokkoli.

Zu Beginn der Filmsequenz dreht sich das Kaninchen in Richtung des Brokkolis und springt zu ihm. Es isst den Brokkoli und dreht sich in Richtung des Frosches. Nachdem es mit der Pfote auf den Boden klopft, quakt der Frosch und springt weg. Das Kaninchen springt zum Briefkasten und öffnet ihn, woraufhin der Frosch zurückkehrt und quakt.

Eigenschaft	Penguin	Bunny
Länge	20 Sekunden	22 Sekunden
Anzahl Objekte	7	5
Anzahl Aktionen	15	17
Skriptlänge	158 Zeilen	95 Zeilen

Tabelle 3.1.: Kerneigenschaften der Filmsequenzen

Die Kerneigenschaften der Filmsequenzen sind noch einmal in Tabelle 3.1 zusammengefasst. Die Alice-Skripte zu beiden Welten befinden sich in Anhang B und C.

3.1.2. Kriterien für die Filmauswahl

Die Kriterien, die von vorneherein als wichtig festgelegt wurden, waren:

- Länge: Die Laufzeit der Filmsequenzen sollte nicht zu lang sein, um Konzentrationsschwierigkeiten und fehlende Bereitschaft zur Teilnahme bei den Testpersonen zu vermeiden
- Anzahl an Objekten: Die Alice-Welten sollten aus nicht zu vielen Objekten bestehen, damit die Testpersonen die Übersicht nicht verlieren
- Quelle: Die Filmsequenzen sollten entweder selbst oder von unabhängigen Personen erstellt worden sein
- Interaktivität: Die Alice-Welten sollten keine Interaktivität beinhalten, sondern wie ein Film ablaufen und vom Benutzer keine Klicks erfordern

Eine der ersten Entscheidungen, die in diesem Auswahlprozess getroffen werden musste, war, ob die Filmsequenzen selbst erstellt oder fertige Alice-Welten aus dem Internet herangezogen werden sollten.

Argumente für fremd erstellte Welten aus dem Internet waren die Neutralität und die nicht vorhandene Beeinflussung. Diese Beeinflussung hätte womöglich bestehen können, wenn es darum geht, Ergebnisse zu erhalten, welche die Analyse vereinfachen. Diese fehlende Neutralität hätte die weitere Verarbeitung der erhaltenen Texte negativ beeinflussen können. Möglicherweise wäre ein falsch positives Ergebnis durch zu einfach zu beschreibende Videos entstanden.

Ein Problem, das bei den existierenden Alice-Welten zu beobachten war, ist die Tatsache, dass diese oftmals interaktiv gestaltet waren. Das bedeutet, dass der Benutzer während die Filmsequenz läuft, deren Ablauf selbst durch Klicks auf verschiedene Objekte beeinflusst. Bei der Verwendung einer solchen Welt wäre das Ergebnis nicht mehr einheitlich gewesen, da möglicherweise verschiedene Testpersonen unterschiedliche Szenarien beschrieben hätten.

Ein weiteres Problem war die Länge der Welten, die im Internet zu finden waren. Die meisten dieser Welten hatten eine Laufzeit von mindestens 60 Sekunden, was die meisten Probanden wohl als zu lange empfunden hätten. Es wäre damit zu rechnen gewesen, dass bei einem längeren Beschreibungstext die Fehlerquote zu hoch würde und damit der Anteil an nicht weiter verwendbaren Texten steigen würde.

Letztendlich ist die Entscheidung dahingehend gefallen, einige Elemente aus den Alice-Welten, die von Dritten erstellt wurden, zu verwenden. Diese Elemente wurden jedoch den eigenen Bedürfnissen angepasst, um die nötige Neutralität zu wahren und keine zu einfachen Abläufe darzustellen. Die Filmsequenzen sollten dennoch vollständig selbst erstellt werden, damit sie die gewünschte Länge und Anzahl an Objekten haben.

3.1.3. Erstellungsprozess

Um eine Filmsequenz mit dem Programm Alice zu erstellen, muss zunächst eine Alice-Welt erstellt werden. Diese besteht aus einem Hintergrund, der mit verschiedenartigen Objekten bestückt wird. Danach können die Objekte bewegt werden und interagieren. Dies wird entweder durch vordefinierte Methoden erreicht, oder durch das Definieren neuer Methoden, die sich aus bereits bestehenden Methoden zu größeren Komplexen zusammenfügen lassen. Ist die Welt mit allen Objekten und den dazugehörigen Methoden, bzw. Aktionen, vollendet, kann sie als Filmdatei exportiert und betrachtet werden.

Als erstes wurde die Alice-Welt „Bunny“ erstellt. Diese sollte ursprünglich die Beispielsequenz werden. Als zweites entstand die Welt „Penguin“, deren entsprechende Sequenz von den Testpersonen beschrieben werden sollte. Diese wurde zuerst etwas schwieriger gestaltet, um zu verhindern, dass die Probanden sich zu stark an der Musterlösung orientieren. Die abweichende Schwierigkeit hätte jedoch ebenfalls die Neutralität des Ergebnisses verletzt und zudem wohl zu teilweise unbrauchbaren Ergebnissen geführt, da die Handlung der Filmsequenz zu kompliziert war. Daraufhin wurde sowohl die Anzahl der Objekte als auch die Anzahl der Aktionen reduziert, bis sie ungefähr gleich der Anzahl im Video „Bunny“ waren. Ebenfalls wurden einige Aktionen, die zuvor gleichzeitig stattfanden, nacheinander ausgeführt, um die Handlung weiter zu entzerren.

Obwohl die Sequenz jetzt einfacher zu beschreiben war, wurde letztendlich beschlossen, dass ein besseres Ergebnis zu erzielen war, wenn die Probanden den Film „Bunny“ statt des Filmes „Penguin“ beschreiben. Die Tatsache, dass im Film „Penguin“ einige Methoden von Methoden auf anderen Objekten aufgerufen werden, ist in der Musterlösung nicht ohne weiteres ersichtlich, denn diese bildet die Methode `world.myFirstMethod()` ab. Diese Methode ist diejenige, die in Alice für den Benutzer sichtbar ist und den Hauptorientierungspunkt für die Animation darstellt. Sie gibt jedoch keine Implementierungsdetails über zusammengesetzte Methoden preis, sondern ruft diese Methoden lediglich auf.

Für den korrekten Ablauf des Filmes ist es jedoch unabdingbar, dass diese Methoden an den im Code definierten Stellen aufgerufen werden und nicht in `world.myFirstMethod()`. Ein Beispiel ist die Methode `monkey.pressButton()`, deren Quellcode folgendermaßen aussieht:

```
monkey.pressButton ()
  Do in order
    Do together
      monkey.right arm turn backward 0,1 revolutions
           duration = 0,5 seconds
      monkey.right arm.right forearm turn forward 0,1 revolutions
           duration = 0,5 seconds
      monkey.right arm.right forearm roll right 0,1 revolutions
           duration = 0,5 seconds
      monkey turn forward 0.03 revolutions duration = 0,5 seconds
    lightBulb.switch
    Do together
      monkey turn backward 0.03 revolutions duration = 0,5 seconds
      monkey.right arm.right forearm turn backward 0,1 revolutions
           duration = 0,5 seconds
      monkey.right arm.right forearm roll left 0,1 revolutions
           duration = 0,5 seconds
      monkey.right arm turn forward 0,1 revolutions
           duration = 0,5 seconds
```

Man sieht, dass die Methode `lightBulb.switch()` etwa in der Mitte dieser Methode aufgerufen wird. Dies hätte bei den Beschreibungstexten dazu geführt, dass diese sowohl von der Musterlösung als auch vom Alice-Skript erheblich abweichen. Im Film „Bunny“ werden keine Methoden innerhalb anderer Methoden aufgerufen. Daher kann eine Beschreibung dieses Videos von den Probanden durchaus fehlerfrei angefertigt werden, ohne zu wissen, welche Methode eine andere aufruft.

Allgemein wurde bei beiden Filmsequenzen darauf geachtet, dass sie beide ungefähr 20 Sekunden lang sind und die gleiche Anzahl an Objekten und Aktionen enthalten. Außerdem wurde darauf geachtet, dass alle Aktionen gut erkennbar und nicht zweideutig sind, damit das Ergebnis damit nicht unnötig kompliziert wird.

Es wurden in beide Videos mehrere Aktionen eingebaut, die aus mehreren einzelnen Methoden zusammengesetzt sind und einen eigenen, beschreibenden Namen haben. Diese Methoden haben den Vorteil, dass sie nur einmal erstellt werden müssen und dann beliebig verwendet werden können. Dies führt zu geringerer Redundanz und ist ein Grundkonzept der objektorientierten Programmierung. Ein Beispiel hierfür ist die Methode `bunny.eat ([Obj] food)`, die das Kaninchen ein Objekt essen lässt. Im Alice-Skript sieht diese Methode wie folgt aus:

```
bunny.eat ([Obj] food)
  Do together
    bunny turn forward 0.15 revolutions duration = 0,5 seconds
    bunny.rightLeg turn backward 0,15 revolutions duration = 0,5 seconds
    bunny.leftLeg turn backward 0,15 revolutions duration = 0,5 seconds
    bunny.upperBody.head turn forward 0,1 revolutions
         duration = 0,5 seconds
  food set isShowing to false
  Do together
    bunny turn backward 0,15 revolutions duration = 0,5 seconds
    bunny.leftLeg turn forward 0,15 revolutions duration = 0,5 seconds
    bunny.rightLeg turn forward 0,15 revolutions duration = 0,5 seconds
    bunny.upperBody.head turn backward 0,1 revolutions
         duration = 0,5 seconds
```

Man kann erkennen, dass der Eindruck, dass das Kaninchen ein Objekt isst, dadurch erreicht wurde, dass das Kaninchen gleichzeitig den gesamten Körper nach vorne neigt, während die Beine sich um den gleichen Winkel nach hinten drehen. Durch die Kombination dieser beiden Bewegungen wurde der Effekt erzielt, dass nur der Oberkörper nach vorne geneigt wird. Anschließend wurde die Sichtbarkeit des übergebenen Objektes auf `false` gesetzt, sodass es so aussieht, als wäre das Objekt verschwunden. Zum Schluss wurden Körper und Beine wieder um den gleichen Winkel gedreht, jedoch in die entgegengesetzte Richtung. Dies bewirkt, dass das Kaninchen wieder in die Ausgangslage zurückkehrt.

Ein weiterer wichtiger Punkt bei der Erstellung der Filmsequenzen war, dass alle Objekte, die während deren Ablauf zu sehen sind, sich schon zu Anfang im Bild befinden. Dies liegt ebenfalls darin begründet, dass die Texte möglichst gut weiter zu verarbeiten sein sollen. Bei der Erstellung einer Alice-Welt müssen alle Objekte schon beim Start der Welt in dieser platziert sein. Ein Objekt kann sich zwar außerhalb des Sichtbereichs des Benutzers befinden, wäre aber technisch gesehen trotzdem schon Teil der Welt. Diese Tatsache wäre einer Testperson, die das Programm Alice nicht kennt, nicht unbedingt bewusst, und sie würde bei der Beschreibung des initialen Aufbaus dieses Objekt wahrscheinlich nicht erwähnen. Würde ihr Beschreibungstext dann in Programmcode umgewandelt, würde dabei

ein Fehler entstehen, denn das Objekt, dass später ins Bild kommen soll, wäre von Anfang an gar nicht in der Welt vorhanden.

3.2. Die Musterlösungen

Zu beiden Filmsequenzen wurden entsprechende Musterlösungen erstellt, welche die zu den Filmen gehörenden Alice-Skripte exakt und lückenlos beschreiben sollen. Eine der Musterlösungen, die des Filmes „Penguin“, diente wie bereits erwähnt den Testpersonen bei der Betrachtung dieses Videos als Anhaltspunkt für die Anfertigung ihres eigenen Beschreibungstextes. Die andere Musterlösung, die des Filmes „Bunny“, wurde hingegen erstellt, um die erhaltenen Texte mit ihr vergleichen zu können.

3.2.1. Zweck

Die Musterlösung für die Sequenz „Penguin“ soll den Testpersonen Ideen für ihre eigene Beschreibung dienen. Ein solcher Anhaltspunkt ist vor allem für Personen ohne Programmierkenntnisse notwendig, da diese mit großer Wahrscheinlichkeit noch nie Programmcode gesehen haben und daher nicht wissen, worauf sie bei einer solchen Aufgabenstellung achten müssen, damit ein Computer ihre Anweisungen interpretieren kann. Es ist weiterhin wichtig, dass die Beschreibungstexte vollständig sind und idealerweise zu jeder Zeile im Skript ein Satz geschrieben werden sollte.

Dabei soll die Musterlösung keineswegs als genaue Anweisung betrachtet werden, was zu tun ist. Bei der Durchführung des Versuches mit den Testpersonen wurden betont, dass das Einbringen eigener Ideen in die Beschreibung durchaus erwünscht ist. Dies ist sogar notwendig, um ein möglichst breites Spektrum an Beschreibungen zu erhalten, die von verschiedenen Menschen erbracht werden könnten. Würde sich jeder strikt an die Musterlösung halten, wäre am Ende das Ergebnis zu einseitig und damit nicht repräsentativ.

Die zweite Musterlösung, die für die Filmsequenz „Bunny“, wurde den Testpersonen nicht vorgelegt, da diese Sequenz von ihnen selbstständig beschrieben werden sollte. Sie wurde jedoch trotzdem erstellt, vordergründig um während der Analysephase eine Vergleichsmöglichkeit mit den erhaltenen Beschreibungstexten zu haben. Dabei geht es besonders darum, zu überprüfen, ob Aktionen vergessen oder Methoden richtig benannt wurden, was dann möglich ist, ohne das Alice-Skript zu betrachten. Der Fließtext hat den weiteren Vorteil, dass er sich schon in der benötigten Form befindet, die von den Beschreibungstexten erwartet wird. Er kann außerdem in weiterführenden Arbeiten als Referenz herangezogen werden, um die Güte einer Beschreibung zu beurteilen.

3.2.2. Linguistischer Aufbau

In der Filmsequenz „Penguin“ gibt es einige Methoden, die von anderen Methoden aus aufgerufen werden, so zum Beispiel die Methode `bucket.knockOver()`, die den Eimer umwirft. Diese wird bei Erfüllung einer bestimmten globalen Variable innerhalb der Methode `penguin.glide()`, bei der der Pinguin über den Boden rutscht, aufgerufen. Ein weiterer Fall ist die Methode `lightBulb.switch()`, welche die Glühbirne je nach Zustand ein- oder ausschaltet, die von `monkey.pressButton()` aufgerufen wird. Diese Tatsache spiegelt sich auch in der Musterlösung wider, in der man weder das Umstoßen des Eimers, noch das An- oder Ausschalten der Glühbirne findet. Dies liegt darin begründet, dass sich die Musterlösung so genau wie möglich an das Alice-Skript halten soll.

Bei den auf Alice-Objekten und -Unterobjekten definierten Methoden kommt es häufig vor, dass beispielsweise Drehungen als Bruch oder die Dauer von Aktionen in Sekundenbruchteilen angegeben werden müssen. Dies ist in der Musterlösung schwierig umzusetzen,

Drehung um	Benennung
1,0	full turn
0,5	half turn
0,25	quarter turn
< 0,25	a small amount

Tabelle 3.2.: Benennung der Drehungen

da natürliche Sprache solche Definitionen zumindest im Alltag nicht kennt, und diese für das menschliche Auge auch kaum exakt zu bestimmen sind. Deshalb wurde für die Musterlösungen eine einheitliche Nomenklatur eingeführt, die, wenn notwendig, eine Drehung mit natürlichen Worten beschreiben kann. Die Benennungen, die diese für bestimmte Brüche vorsieht, sind in Tabelle 3.2 zu sehen.

Ein Beispiel für eine solche Drehung in der Filmsequenz „Bunny“ ist die folgende Stelle:

> The bunny's head turns forward by a small amount (...).

Der Kopf des Kaninchen wird im zugehörigen Programmcode um 0,15 Umdrehungen nach vorne gedreht.

Da die Dauer einer Aktion für das menschliche Auge kaum zu bestimmen ist, wurde lediglich für eine Aktionsdauer von weniger als einer Sekunde der Zusatz *quickly* an das jeweilige Verb angefügt, so zum Beispiel in der Musterlösung für die Sequenz „Penguin“:

> The duck prince quickly turns to face the penguin.

Die Aktionsdauer beträgt an dieser Stelle im Programm 0,2 Sekunden.

Die Aussagen *A very short time passes*, die sich an mehreren Stellen in beiden Texten befinden, entsprechen im Programmcode einem Warten von 0,5 Sekunden. Es war einkalkuliert, dass die Probanden diese Pausen in der Filmsequenz bei der Anfertigung ihrer Beschreibung nicht erkennen würden. Die Pausen wurden lediglich eingebaut, um den Film zu entzerren und optisch ansprechender zu machen. In der Musterlösung fanden diese Stellen Erwähnung, weil die Musterlösung den Code korrekt abbilden sollte, was nur möglich ist, wenn kein Befehl ausgelassen wird.

3.2.3. Gegenüberstellung von Musterlösung und Programmcode

In Tabelle 3.3 werden das Alice-Skript der Welt „Penguin“ und die dazugehörige Musterlösung einander gegenübergestellt. So lässt sich nicht nur der linguistische Aufbau des Textes besser verstehen, sondern auch das Verhältnis der Anzahl aufgerufener Methoden zur Anzahl Sätze in der Musterlösung ablesen.

Im folgenden soll der Fall der Filmsequenz „Penguin“ betrachtet werden. Die folgende Tabelle enthält in der ersten Spalte die Zeilennummer des Quelltextes, in der zweiten Spalte den Code, der direkt aus dem Programm Alice extrahiert und exakt übernommen wurde und in der dritten Spalte schließlich den entsprechenden Satz in der Musterlösung (sofern vorhanden).

Als erstes fällt auf, dass jeder Aktion, die in der Filmsequenz abläuft, eine Zeile im Code entspricht. Dementsprechend gibt es auch zu jeder Zeile im Code einen Satz in der Musterlösung. Die einzige Ausnahme bildet die erste Zeile, bei der lediglich eine Sekunde gewartet wird. Diese Aktion stellt keine wichtige Handlung dar, sondern wurde nur eingefügt, damit die Objekte bei Start des Videos nicht gleich beginnen, sich zu bewegen. Daher wurde diese Wartezeit nicht im Beschreibungstext erwähnt.

Zeile	Alice-Skript	Musterlösung
1	`Wait 1 second`	-
2	`DuckPrince turn to face monkey`	The duck prince turns to face the monkey.
3	`DuckPrince.command`	The duck prince commands.
4	`monkey.sigh`	The monkey sighs.
5	`monkey turn to face Remote`	The monkey turns to face the remote.
6	`monkey.jump`	The monkey jumps.
7	`monkey.pressButton`	The monkey presses a button.
8	`Wait 0,5 seconds`	A very short time passes.
9	`penguin turn to face bucket`	The penguin turns to face the bucket.
10	`penguin.glide`	The penguin glides.
11	`DuckPrince turn to face penguin duration = 0,2 seconds`	The duck prince quickly turns to face the penguin.
12	`DuckPrince.scold`	The duck prince scolds.
13	`Wait 0,5 seconds`	A very short time passes.
14	`monkey.laugh`	The monkey laughs.
15	`monkey.pressButton`	The monkey presses a button.
16	`penguin.jump times = 1`	The penguin jumps once.

Tabelle 3.3.: Vergleich des Codes und der Beschreibung zur Sequenz „Penguin“

Filmsequenz	Sätze	Wörter	Zeichen	Vokabulargröße
Bunny	21	157	847	51
Penguin	23	157	874	48

Tabelle 3.4.: Kerneigenschaften der Musterlösungen

Es fällt auf, dass in Zeile 16 der Parameter `times = 1` beim Aufruf der Methode `penguin.jump` übergeben wird. Dies ist sonst nicht der Fall und liegt daran, dass die Methode `jump` in Alice vordefiniert war. Dies ist verständlicherweise für bearbeitende Testpersonen schwer durchschaubar, musste aber in der Musterlösung berücksichtigt werden, weil sonst bei einer möglichen späteren Übersetzung in Code der Methode kein Parameter übergeben würde und dies zu einem Übersetzungsfehler führen könnte.

Ein weiterer wichtiger Punkt ist, dass im Beschreibungstext alle Aktionen mit ihrem entsprechenden Methodennamen benannt wurden. Dies trifft ebenfalls auf die Namen der Objekte zu. Auch diese Entscheidung zielt auf die Übersichtlichkeit und leichtere Interpretierbarkeit durch den Compiler ab, eine Eigenschaft, die bei der weiteren Bearbeitung der gesammelten Texte von großer Bedeutung ist. Daher wurde auf diesen Punkt in der Aufgabenstellung, die im nächsten Unterkapitel besprochen wird, besonders Wert gelegt.

Der erste Teil der Musterlösung, der mit weiteren acht Sätzen rund ein Drittel des gesamten Textes ausmacht, ist in dieser Tabelle nicht aufgeführt, da er den initialen Aufbau der Szenerie und Objekte beschreibt. Da dieser im Alice-Skript nicht berücksichtigt ist, wurde der erste Teil der Musterlösung in der Tabelle außen vor gelassen, da ein Vergleich nicht möglich gewesen wäre.

Abschließend bleibt zu sagen, dass die Musterlösungen beide etwa gleich lang sind, was sich an der Statistik in Tabelle 3.4 ablesen lässt.

Filmsequenz	Sätze in der Musterlösung	Anweisungen im Skript
Bunny	21	17
Penguin	23	15

Tabelle 3.5.: Musterlösungen im Vergleich mit Anweisungen im Skript

In Tabelle 3.5 ist die Anzahl der Sätze in beiden Musterlösungen der Anzahl der Anweisungen im Alice-Skript der dazugehörigen Filmsequenzen gegenübergestellt. Dass die Anzahl der Sätze in der Musterlösung in diesem Fall nicht mit der Anzahl der Anweisungen übereinstimmt, hängt damit zusammen, dass in der Musterlösung zusätzlich der Aufbau der Welt beschrieben ist.

Die komplette Musterlösung zu beiden Filmsequenzen, einschließlich der Beschreibungen des Aufbaus der Welten, befinden sich in Anhang D.

3.3. Aufgabenstellung

Der nächste Schritt nach der Erstellung der Videos war die Überlegung, wie die genaue Aufgabenstellung an die Probanden aussehen sollte. Dabei war es wichtig, den Probanden eine Anleitung zu geben, die möglichst konsistent sein sollte, um am Ende ein homogenes Ergebnis zu erhalten. Da nicht alle Testpersonen ihre Beschreibung gleichzeitig anfertigen würden, war zu befürchten, dass bei einer mehrmaligen mündlichen Erklärung der Aufgabenstellung einzelne Probanden benachteiligt werden könnten, wenn einige Anweisungen vergessen würden. Auch das hätte zur Folge gehabt, dass die Texte nicht repräsentativ sind, was das Ergebnis dieser Arbeit negativ beeinflusst hätte.

Es war also notwendig, eine schriftliche Anleitung zu formulieren, die es auch erlaubt, dass Testpersonen die Aufgabe ohne mündliche Anweisungen bearbeiten könnten. Aus dieser Notwendigkeit entstand ein kompletter Fragebogen. Dieser enthält zunächst eine Einleitung in die Problemstellung und Fragen zu persönlichen Informationen, die zur späteren Analyse der Texte dienten. Sie sollten zum Beispiel darüber Auskunft geben, ob die Testperson Programmierkenntnisse besitzt oder nicht. Im nächsten Schritt erhielten die Testpersonen mit der Filmsequenz „Penguin“ ein erstes Beispiel, wobei ihnen hier auch die Musterlösung vorgelegt wurde.

Anschließend sollte dann die Filmsequenz „Bunny“ beschrieben werden. Hierzu wurden den Probanden die Objekte mit ihren Namen vorgestellt, die sie später bei der Anfertigung ihrer Beschreibung verwenden sollten. Dies sollte gewährleisten, dass die Beschreibungstexte einheitlich würden und die gleichen Objektnamen verwenden würden. Dies dient der einfacheren Weiterverarbeitung im späteren Projektverlauf. Außerdem wurden den Probanden die Methoden der Objekte in tabellarischer Form aufgelistet, um auch hier eine einheitliche Benennung zu ermöglichen.

Die Beschreibung sollte im Übrigen bevorzugt auf Englisch angefertigt werden. Dies hat zwei Gründe. Ein Grund ist, dass Alice mit dieser Sprache arbeitet. Der zweite Grund ist der Vorteil, der für die Weiterverarbeitung der Texte entsteht. Die meisten Werkzeuge, die für die Verarbeitung von natürlichsprachlichen Texten existieren, arbeiten nur mit der englischen Sprache. Es war den Probanden trotz allem gestattet, die Beschreibung auf Deutsch anzufertigen, da davon auszugehen war, dass einige Probanden nicht über ausreichende Englischkenntnisse verfügten. Vor der Analyse wurden die betroffenen Texte jedoch auf Englisch übersetzt.

3.4. Die Probanden

Bei der Auswahl der Probanden war vor allem wichtig, dass sowohl Personen mit Programmierkenntnissen als auch Personen ohne Programmierkenntnisse ausgewählt werden. Diese beiden Gruppen wurden bei dieser Arbeit etwa gleich groß gehalten, um beide Seiten angemessen betrachten zu können.

Von Personen mit Programmierkenntnissen wurde erwartet, dass sie eine Beschreibung liefern, die sich im weiteren Projektverlauf gut weiterverarbeiten lässt. Das bedeutet, dass die Beschreibung sowohl vollständig ist, als auch sprachlich von einem Computer fehlerfrei interpretiert werden kann. Hier sind die Probanden mit objektorientierten Programmierkenntnissen im Vorteil, denn sie haben bei der Betrachtung des Videos eine Vorstellung davon, wie der dazugehörige Quellcode aussehen könnte, welche Objekte existieren und wie diese miteinander kommunizieren. Auf der Basis dieses Wissens könnten sie bestenfalls versuchen, diesen Code in natürlicher Sprache zu rekonstruieren und somit zu einem sehr guten Ergebnis zu kommen.

Die Probanden ohne Programmierkenntnisse sind diejenigen, für die das System am Ende entwickelt werden soll. Deswegen sollte man sie bei der Sammlung von Texten auf jeden Fall mit berücksichtigen. Es war zu erwarten, dass die Beschreibungstexte dieser Personen im Durchschnitt nicht den gleichen Anforderungen gerecht würden wie die der Personen mit Programmierkenntnissen. Da Mitglieder dieser Testgruppe nicht programmieren können und vorher auch wahrscheinlich noch nie Quellcode gesehen haben, wussten sie im Allgemeinen nicht, was sie bei der Erstellung ihrer Beschreibung beachten müssen. Dies könnte dazu führen, dass subtile Teile der Handlung nicht erkannt oder nicht beachtet werden, da die Probanden die Bedeutung dieser Details für den Computer nicht erkennen. Ebenfalls kann es dazu führen, dass der Satzbau für eine Interpretation durch den Compiler nicht geeignet ist.

3.5. Die Durchführung

Bei der Durchführung müssen ebenfalls zwei Gruppen von Probanden unterschieden werden. Gruppe A enthält diejenigen, welche den Fragebogen unter Aufsicht und mit zusätzlicher mündlicher Instruktion ausgefüllt haben. Gruppe E enthält diejenigen, welche die Bearbeitung selbstständig und ohne weitere Anweisungen durchgeführt haben.

Insgesamt betrug die Anzahl der Personen in Gruppe A sechs. Diesen Testpersonen musste teilweise genauer erläutert werden, welches System entwickelt werden soll und was unsere Ziele sind. Viele der Probanden in dieser Gruppe hatten keine Programmierkenntnisse. Es war nicht davon auszugehen, dass sie sich vorstellen können, welche Schritte im Detail benötigt werden, um ein solches System zu entwickeln. Trotzdem war es notwendig, ihnen dieses System zu erklären, um einen guten Beschreibungstext zu erhalten, da die Probanden erst dann wüssten, wo die Prioritäten in der Beschreibung liegen.

Bei dieser Gruppe wurden auch während der Anfertigung des Textes noch viele Fragen gestellt und beantwortet. Darunter befanden sich zum Beispiel Fragen, ob auch die Farben der Objekte beschrieben werden müssen, ob der Aufbau der Szenerie wirklich beschrieben werden muss, und ob man aufschreiben muss, wie oft das Kaninchen springt. Diese Fragen zeugen nicht unbedingt von Unwissen aufgrund von fehlenden Programmierkenntnissen, sie zeigen auch, dass das Programm Alice nicht ausnahmslos leicht zu durchschauen ist, wenn man es noch nie bedient hat. Dass die Farben von Objekten nicht festgelegt werden müssen, sollte den Testpersonen in zukünftigen Anleitungen erklärt werden, falls eine weitere solche Studie stattfindet. Ob bei einer Methode ein Parameter für die Häufigkeit der Ausführung übergeben werden muss, kann man nur wissen, wenn man das Programm selbst bedient

hat und den Quellcode der Methoden kennt. Ein Merkmal, das nur in dieser Gruppe beobachtet werden konnte, war die benötigte Bearbeitungszeit für den Fragebogen. Diese war sehr unterschiedlich und variierte zwischen 10 und 60 Minuten.

In Gruppe E, die ihre Fragebögen eigenständig und ohne weitere mündliche Instruktion ausfüllte, gehörten sowohl Personen mit Programmierkenntnissen als auch Personen ohne Programmierkenntnisse an. Es war insgesamt zu beobachten, dass die meisten Probanden es vorzogen, den Fragebogen ohne Aufsicht auszufüllen, vorzugsweise am eigenen Arbeitsplatz, daher war diese zweite Gruppe auch etwas größer. Hier konnten verständlicherweise keine weiteren Fragen mehr gestellt werden, wonach zu erwarten war, dass die Qualität der Texte, zumindest bei Personen ohne Programmierkenntnisse, schlechter ausfallen würde als bei der ersten Gruppe. Es stellte sich jedoch heraus, dass es genau umgekehrt war, worauf ich im nächsten Kapitel genauer eingehen werde.

Eine weitere Beobachtung während der Durchführung war, dass die Probanden die Texte sehr ungern auf Englisch verfassen wollten. Es war ihnen freigestellt, diese auf Englisch oder Deutsch zu verfassen.

4. Vergleich und Analyse der Texte

Es ist während dieser Arbeit ein Alice-Korpus aus 14 Beschreibungstexten entstanden, der für weitere Arbeiten dienen kann. Es kann sowohl auf der Grundlage dieser Texte weitergearbeitet werden, als auch mit Hilfe der Filmsequenzen das Korpus bei Bedarf erweitert werden. In Tabelle 4.1 finden sich die Kennzahlen der Texte, das heißt die jeweilige Anzahl Sätze, die Textlänge, die Zeichenanzahl und die Vokabulargröße. Außerdem finden sich in dieser Tabelle Informationen über die Verfasser der Texte, nämlich ob diese Programmierkenntnisse hatten oder nicht und ob sie die Texte unter Aufsicht verfasst haben oder nicht.

Es gab insgesamt sechs Probanden ohne Programmierkenntnisse und acht Probanden mit Programmierkenntnissen. Sechs Probanden haben ihre Texte unter Aufsicht verfasst, acht Probanden waren bei der Anfertigung der Texte alleine. Auf diese Zahlen wird in Unterkapitel 4.2 genauer eingegangen. Es ist während dem Aufbau des Korpus kein Text verloren gegangen und auch kein Text doppelt vorhanden.

In diesem Kapitel werden die erhaltenen Beschreibungstexte sprachlich analysiert und miteinander verglichen. Zuerst werden die Texte linguistisch betrachtet. Dann werden die Wort- und Satzanzahl der Texte bestimmt, um zu sehen, wie sich diese im Vergleich zur Musterlösung verhalten. Danach werden die Texte mit einem automatischen Wortart-Markierer annotiert, um zu sehen, ob sie sprachlich so korrekt sind, dass die Wortarten und Grammatik automatisch erkannt werden können.

4.1. Linguistische Analyse

Allgemein ist beim Betrachten der Texte zu sehen, dass diese bis auf wenige Ausnahmen vollständig sind und keine Aktionen fehlen oder übersehen wurden. Auch die verwendete Sprache ist weitestgehend verständlich.

Allerdings muss man auch beobachten, dass die Vorgaben der Anleitung nicht immer eingehalten wurden. So wurden die Angaben der Objektnamen oft ignoriert, wie zum Beispiel in Text 3 (Proband ohne Programmierkenntnisse):

> Left background with palm.

Der Unterschied zwischen palm und PalmTree mag nicht groß erscheinen, doch PalmTree ist hier der Name des Objektes. Eine Verwendung würde die Weiterverarbeitung vereinfachen, da das Objekt sofort erkannt würde und keine Interpretation stattfinden müsste.

Es war zu erwarten, dass vor allem Probanden ohne Programmierkenntnisse diese Art von Fehler machen würden, doch dies war nicht der Fall. Ein Beispiel aus Text 10 belegt, dass auch Probanden mit Programmierkenntnissen sich nicht an die Anleitung hielten:

> Behind the mailbox there is a palm.

Außerdem muss teilweise davon ausgegangen werden, dass nicht nur die Anweisung ignoriert oder vergessen wurde, sondern die Seite mit den Objektnamen nicht einmal gelesen wurde, wie ein Beispiel aus Text 7 vermuten lässt:

> Before the Bunny stands a mushroom.

Bei dem hier mit *mushroom* (Pilz) bezeichneten Objekt handelt es sich um einen Brokkoli, was auf der zweiten Seite des Fragebogens erwähnt wurde. Es kann ebenfalls beobachtet werden, dass ein Objektname (in diesem Beispiel `Bunny`) sich während der Beschreibung im Text plötzlich ändert, wie in Text 11:

> The rabbit eats the broccoli. (...) The bunny hops to the mailbox, (...).

Weiter wurden im Fragebogen die Methodennamen vorgegeben, um den Probanden die Beschreibung zu erleichtern und die Texte zu vereinheitlichen. Auch diese Vorgabe wurde von einigen Probanden nicht eingehalten, was die folgenden beiden Beispiele verdeutlichen:

> Frog says something to bunny.

Dieses Zitat stammt aus Text 7 und bezeichnet die Stelle `frog.ribbit()`. Die Methode `ribbit` war in der Tabelle auf Seite 2 des Fragebogens für das Objekt `frog` angegeben. Auch in Text 12 findet sich eine entsprechende Stelle:

> The Bunny turns towards the frog.

Hier handelt es sich jedoch um einen geringfügigen Fehler, da die korrekte Methode `turn to face` heißt. Diese Textstellen verdeutlichen, dass das System später wohl eine größere Vielfalt an sprachlichen Formulierungen interpretieren können und diese dann auf Alice-Methoden abbilden muss.

4.2. Wort- und Satzanzahl

Um die Wort- und Satzanzahl zu betrachten, wurde ein Shell-Skript erstellt, welches diese Aufgabe automatisch erledigt. Dies dient nicht nur dazu, die Schreibarbeit zu verringern, sondern auch der Erweiterbarkeit, da bei Veränderung der Texte die Befehle nicht neu eingegeben werden müssen, sondern lediglich das Skript neu ausgeführt werden muss. Der hierbei verwendete Befehl war `wc` (word count).

Durch seine Ausführung wurden für die Texte die in Tabelle 4.1 angegebenen Zahlen für die Anzahl Sätze, Wörter und Zeichen ermittelt. Außerdem gibt die Tabelle Auskunft darüber, ob der Proband, der den Text angefertigt hat, Programmierkenntnisse hat, und ob er den Text unter Aufsicht angefertigt hat oder nicht.

Diese Daten wurden nun, damit sie leichter verglichen werden können, mit Hilfe von Gnuplot in sogenannte Boxplot-Diagramme (auch Kastengrafik genannt) übertragen. In Abbildung 4.1 sieht man ein Beispiel für ein solches Diagramm. In der Box liegen die mittleren 50% der Daten. Der Strich stellt den Median dar. Das obere und untere Ende der Box repräsentieren jeweils das obere beziehungsweise untere Quartil[1]. Die Länge der Box ist der

[1] Oberes Quartil bedeutet, dass 75% der Werte kleiner als dieser Wert sind; unteres Quartil bedeutet, dass 25 % der Werte kleiner als dieser Wert sind.

Text	Progr.kenntnisse	unter Aufsicht	Sätze	Wörter	Zeichen
1	ja	ja	20	168	903
2	nein	ja	9	104	509
3	nein	nein	19	164	901
4	nein	ja	11	105	558
5	nein	ja	16	119	628
6	nein	nein	12	120	618
7	nein	nein	14	223	1159
8	ja	nein	23	162	889
9	ja	nein	19	149	820
10	ja	ja	11	106	544
11	ja	ja	11	106	564
12	ja	nein	22	146	833
13	ja	nein	12	125	661
14	ja	nein	13	155	852

Tabelle 4.1.: Anzahl Sätze, Wörter und Zeichen in den Beschreibungstexten, sowie Merkmale der Probanden

Interquartilsabstand. Die Antennen können, vom Rand der Box aus gesehen, maximal 1,5 mal so lang wie der Interquartilsabstand sein. Ihr Ende liegt bei dem größten beziehungsweise kleinsten Wert, der außerhalb der Box, aber noch innerhalb dieses Wertebereiches liegt. Werte, die sich außerhalb des Wertebereiches befinden, werden als Ausreißer betrachtet und mit einem Punkt markiert.

Man kann nun die schon im vorangegangenen Kapitel vorgestellte Gruppenaufteilung heranziehen, um die quantitativen Eigenschaften der Beschreibungstexte unter verschiedenen Gesichtspunkten miteinander zu vergleichen.

4.2.1. Aufteilung nach Programmierkenntnissen

Das Diagramm in Abbildung 4.2 zeigt die Verteilung der Satzanzahl in den Beschreibungstexten. Dabei zeigt der linke Kasten die Verteilung der Satzanzahl bei den Probanden mit Programmierkenntnissen, der rechte Kasten die Verteilung bei den Probanden ohne Programmierkenntnisse.

In Abbildung 4.2 kann man erkennen, dass die Probanden mit Programmierkenntnissen, im folgenden Gruppe P genannt, im Durchschnitt drei Sätze mehr geschrieben haben als die Probanden ohne Programmierkenntnisse (Gruppe O). Dies entspricht genau den Erwartungen. Gruppe O tendierte eher dazu, längere Sätze mit vielen Nebensätzen zu bauen. Dagegen war bei Gruppe P häufiger zu beobachten, dass kürzere Sätze geschrieben wurden. Dies zeigt sich auch gut bei Betrachtung des Diagrammes in Abbildung 4.3, das die durchschnittliche Anzahl an Wörtern pro Satz zeigt.

Diese Tatsache lässt sich vermutlich darauf zurückführen, dass Personen mit objektorientierten Programmierkenntnissen wissen, wie der entsprechende Quellcode aussehen könnte und daher eher kurze, prägnante Anweisungen formulieren, die alle nötigen Informationen enthalten. Dagegen formulieren Personen ohne Programmierkenntnisse ihre Sätze eher umfangreicher und bildlicher. Der folgende Satz stammt aus Text 3 und wurde von einem Probanden ohne Programmierkenntnisse verfasst:

> Bunny turns a bit right to face the mailbox and hops 3 times to left part of the screen and sits down in front of the mailbox.

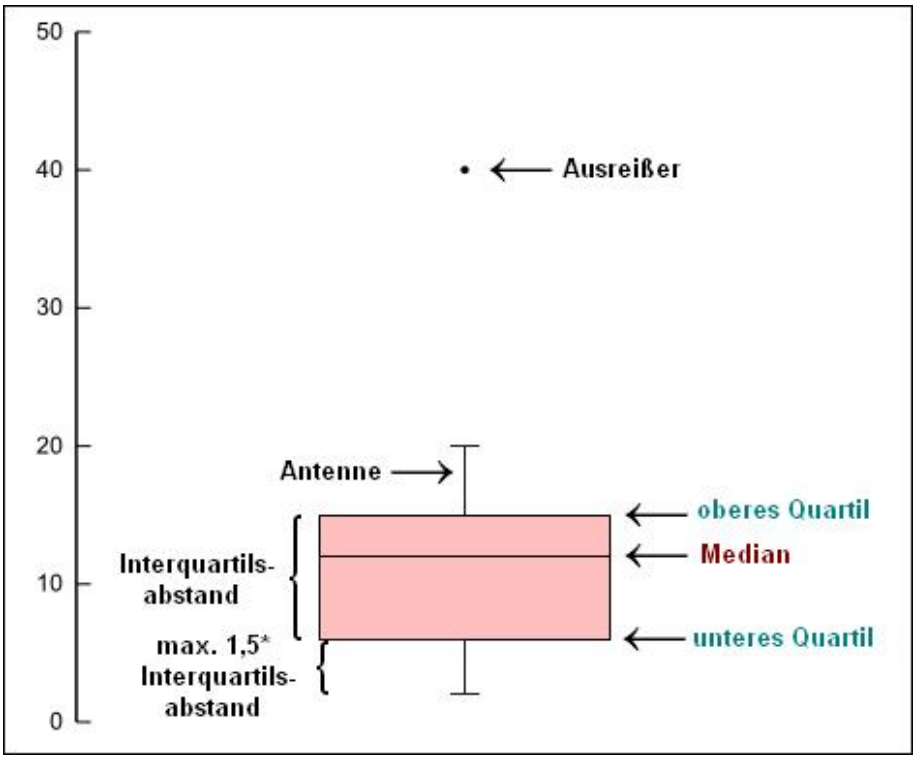

Abbildung 4.1.: Beispiel für ein Boxplot-Diagramm

Der selbe Sachverhalt wurde in Text 1 von einem Probanden mit Programmierkenntnissen folgendermaßen beschrieben:

> The Bunny turns to face the Mailbox. The Bunny hops three times to the Mailbox.

Während Proband 3 für seine Beschreibung einen Satz und 27 Wörter benötigt hat, teilte Proband 1 seine Beschreibung in zwei Sätze und benötigte nur 15 Wörter. Es ist zu erwarten, dass für einen Computer die zweite Version leichter zu verarbeiten ist, da bei weniger Wörtern weniger Mehrdeutigkeiten entstehen können und ein Subjekt-Prädikat-Objekt-Satz ohne Adverbien und Modalpartikel einer der einfachsten grundlegenden Sätze ist, die in einer natürlichen Sprache vorkommen können.

Einige Sachverhalte wurden in Gruppe O möglicherweise übersehen oder nicht als wichtig erachtet. Ein Beispiel hierfür ist die letzte Aktion in der Filmsequenz, `frog.ribbit()`. Diese Aktion wurde bis auf eine Ausnahme von allen Personen in Gruppe P beschrieben, während nur eine Person aus Gruppe O diese Aktion erwähnt hat. Oft wurden auch Teile des initialen Aufbaus nicht beschrieben. So beschreibt zum Beispiel eine Probandin ohne Programmierkenntnisse in Text 2 den gesamten Aufbau:

> I see a palm tree on the left of the screen, a mailbox in front of it. In the foreground there sits a frog on the left and a hare on the right of the screen.

Hier wird weder der Brokkoli erwähnt, noch beschrieben, in welche Richtung die Objekte schauen. Wenn diese Informationen fehlen, ist dies bei einer späteren Bedienung des Systems natürlich problematisch. Werden Objekte der Welt zu Beginn nicht hinzugefügt, später aber erwähnt, wenn sie Aktionen ausführen sollen, oder Methoden auf ihnen aufgerufen werden, führt dies mit großer Wahrscheinlichkeit zu einem Übersetzungsfehler. Auf einen solchen Fehler könnte ein Benutzer entweder mit einer Fehlermeldung hingewiesen

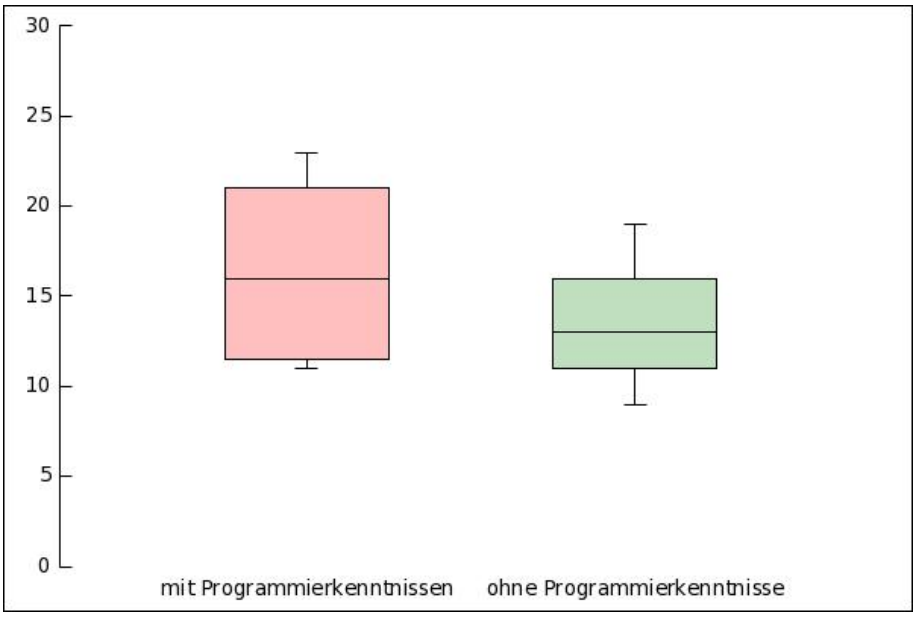

Abbildung 4.2.: Verteilung der Satzanzahl bei den beiden Probandengruppen

werden, oder ein fehlendes Objekt könnte an irgendeine Stelle in der Welt automatisch hinzugefügt werden.

Andere Auslassungen sind weniger schlimm, so zum Beispiel fehlende Drehungen der Figuren oder nicht aufgerufene Methoden, sofern sie keine später notwendige Zustandsänderung auf einem Objekt hervorrufen. Diese Auslassungen hätten lediglich zur Folge, dass das Programm nicht das gewünschte Verhalten zeigt. Bei einem Programm wie Alice wäre ein solcher Fehler dank der grafischen Ausgabe sofort vom Benutzer zu erkennen und könnte ohne große Probleme beseitigt werden.

Es wurde ebenfalls ein Diagramm erstellt, das die Verteilung der Anzahl an Zeichen in den Beschreibungstexten in beiden Gruppen gegenüberstellt (Abbildung 4.4). Diese Anzahl hängt natürlich, da der Rahmen eng vorgegeben war, stark mit der Textlänge zusammen, und so ist es auch nicht verwunderlich, dass sich die beiden entsprechenden Diagramme stark ähneln. Auch hier ist wieder zu beobachten, dass die durchschnittliche Anzahl an Zeichen in Gruppe P höher lag als in Gruppe O, während jedoch die Streuung in Gruppe O minimal größer ist.

Zuletzt zeigt das Diagramm in Abbildung 4.5 die Verteilung der Vokabulargröße. Während der Median bei beiden Gruppen fast gleich ist (Gruppe P: 46, Gruppe O: 44,5), ist die Streuung bei Gruppe O jedoch wesentlich größer. Man sollte bei der Weiterverarbeitung beachten, dass Personen ohne Programmierkenntnisse anscheinend ein größeres Vokabular verwenden als Personen mit Programmierkenntnissen. Daher benötigt man bei der Entwicklung eines Systems, das auf die Verwendung von Personen ohne Programmierkenntnisse ausgelegt ist, mehr Wörterbuchwissen.

4.2.2. Aufteilung nach Beaufsichtigung bei der Bearbeitung

Neben der Aufteilung nach den Programmierkenntnissen wurde im vorangegangenen Kapitel auch erwähnt, dass einige Probanden den Fragebogen unter Aufsicht ausgefüllt haben und die Möglichkeit hatten, während der Bearbeitung Fragen zu stellen. Diese Gruppe wird im Folgenden als Gruppe A bezeichnet. Dagegen bearbeitete Gruppe E den Fragebogen selbstständig. Es bleibt zu erwähnen, dass Gruppe A aus insgesamt sechs Personen

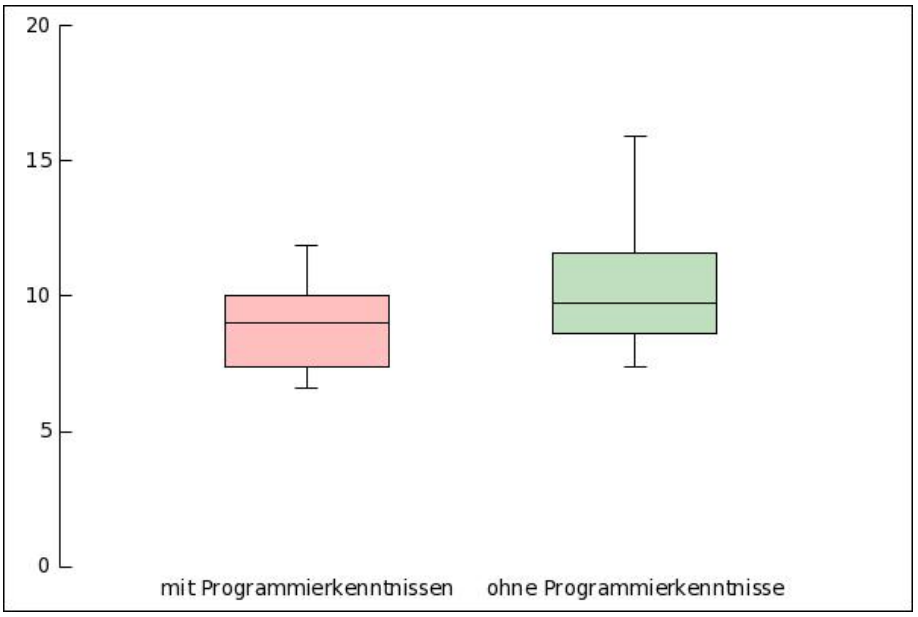

Abbildung 4.3.: Verteilung der durchschnittlichen Anzahl an Wörtern pro Satz

bestand, von denen die Hälfte keine Programmierkenntnisse besaß. Gruppe E bestand aus acht Personen, von denen drei keine Programmierkenntnisse hatten.

Zum Vergleich der Verteilung der Satzzahl betrachten wir wieder das zugehörige Boxplot-Diagramm in Abbildung 4.6. Hier zeigt sich eine Überraschung, denn es war nicht zu erwarten, dass Gruppe A im Schnitt weniger schreibt als Gruppe E. Es wäre eher zu erwarten gewesen, dass Gruppe A qualitativ bessere Texte anfertigen würde als Gruppe E, was sich, wie im letzten Abschnitt erläutert, unter anderem in der Satzzahl niederschlägt.

Auch bei der Textlänge bietet sich das gleiche Bild, wobei diese bei Gruppe A sogar deutlich geringer ist als bei Gruppe E. Die höchste Anzahl an Wörtern, die in Gruppe A geschrieben wurde, liegt unter der geringsten Anzahl an Wörtern, die in Gruppe E geschrieben wurde. In diesem Boxplot-Diagramm (Abbildung 4.7) wird bei Gruppe A ein Ausreißer nach oben angezeigt. Ein Grund für diesen Ausreißer wird später in diesem Abschnitt erläutert, wenn das Ergebnis ausgewertet wird. Auch die Anzahl an Wörtern pro Satz ist etwas geringer, wie in Abbildung 4.8 zu sehen ist, wobei der Median hier in beiden Gruppen etwa gleich ist. Das gleiche Bild bietet sich auch in den letzten beiden Diagrammen (Abbildungen 4.9 und 4.10), welche die Verteilung der Zeichenmenge und der Vokabulargröße zeigen.

Ein Grund für diese angenommene geringere Qualität mag im höheren Anteil an Probanden ohne Programmierkenntnisse in Gruppe A zu finden sein (50% gegenüber 38%). Ein anderer Grund könnte die Ablenkung gewesen sein, die durch die Anwesenheit einer beaufsichtigenden Person oder durch die Bearbeitung in der Gruppe bestand. Es war zu beobachten, dass die Probanden bis auf eine Ausnahme während der Bearbeitung untereinander redeten. Bei dem oben erwähnte Ausreißer nach oben, der auf einigen Diagrammen zu erkennen war, handelte es sich um die einzige Person, die sich während der Bearbeitung nicht mit jemandem unterhielt.

Wenn man die Ergebnisse dieses Abschnitts betrachtet, sollte man, wenn man längere Beschreibungstexte mit kürzeren Sätzen möchte, der anfertigenden Person keine zusätzlichen mündlichen Instruktionen geben. Die Personen sollten sich dabei konzentrieren und nicht abgelenkt werden, um einen Text mit diesen Merkmalen anzufertigen.

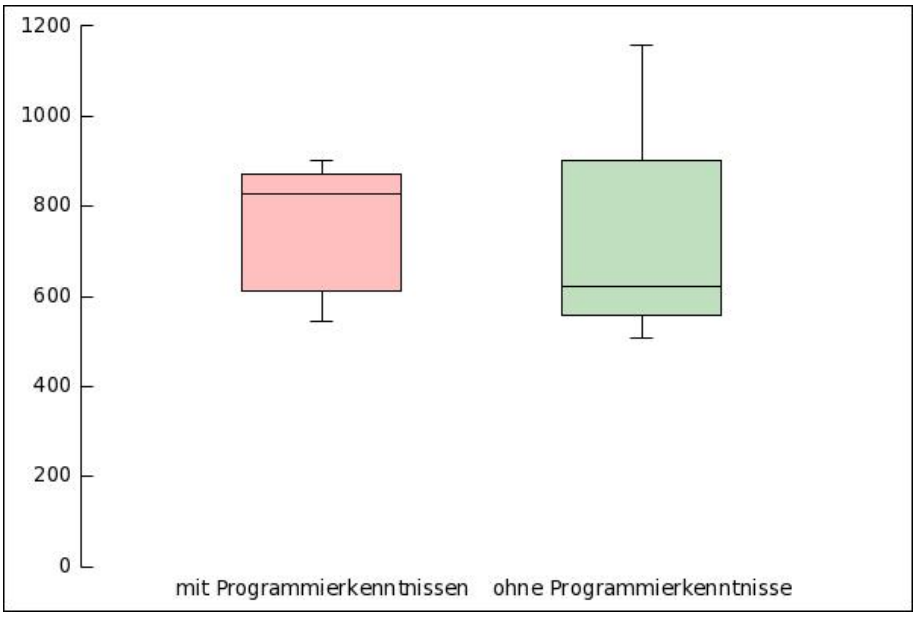

Abbildung 4.4.: Verteilung der Anzahl an Zeichen

Etikett	**Wortart**
NN	Substantiv Singular
NNP	Eigenname (*proper noun*)
VBZ	Verb im Präsens (3. Person Singular)
VBD	Verb in der Vergangenheit
VBG	Gerundium
IN	Präposition
DT	Determinierer
JJ	Adjektiv
EX	*existential there* (Existenz anzeigendes *there*)
PRP$	Personalpronomen
.	Punkt

Tabelle 4.2.: Etiketten und damit bezeichnete Wortarten des University of Illinois Wortart-Markierers [ill]

4.3. Markieren der Wortarten

Um zu überprüfen, ob die Texte sprachlich korrekt und damit gut weiterverarbeitbar sind, wurden sie schließlich annotiert, um die vorkommenden Wortarten zu bestimmen. Für diese Aufgabe wurde der Wortart-Markierer (Part-of-Speech-Tagger) der Universität von Illinois in Urbana-Champaign [ill] verwendet.

Ein Beispiel für eine fehlerfreie Interpretation ist der Satz *In the background on the left hand side there is a PalmTree.* aus Text 1 (vergleiche Anhang E). Der Wortart-Markierer annotiert diesen Satz folgendermaßen:

> IN/In DT/the NN/background IN/on DT/the JJ/left NN/hand NN/side EX/there VBZ/is DT/a NNP/PalmTree ./.

In Tabelle 4.2 sieht man, welche Wortarten durch welche Etiketten angezeigt werden.

Die Wortarten sind zwar alle korrekt klassifiziert, jedoch lassen sich zwei Besonderheiten feststellen. Dass *PalmTree* als Eigenname klassifiziert wird, obwohl es eigentlich ein Sub-

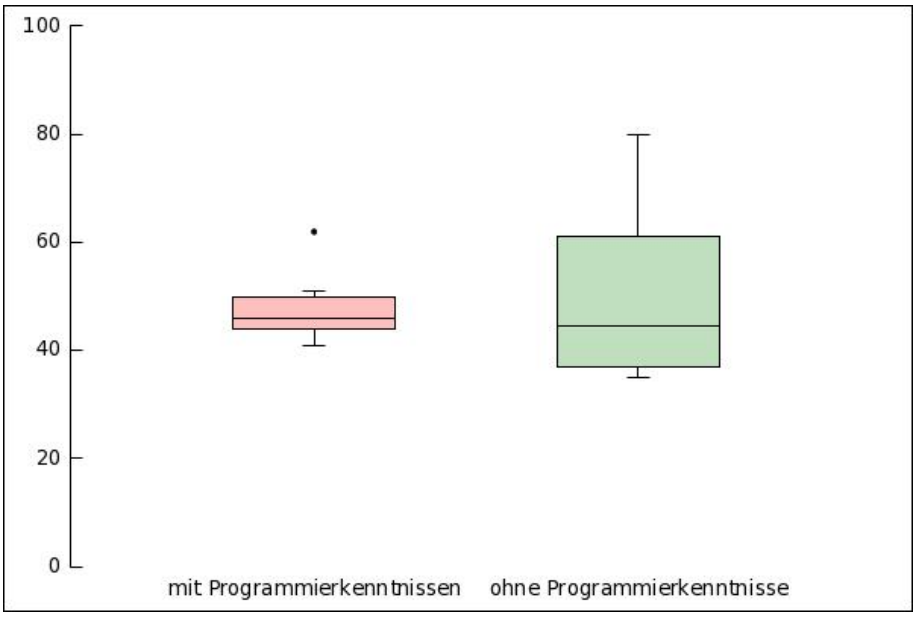

Abbildung 4.5.: Verteilung der Vokabulargröße

stantiv darstellt, liegt daran, dass das Wort groß geschrieben ist. Das ist insofern praktisch, als dass die Palme ein Objekt in der Alice-Welt darstellt. In einem trivialen System könnte das ein Vorteil sein. Ignoriert man allerdings den Objektnamen aus der Liste und schreibt `palm tree` korrekt, so wird es als normales Substantiv gekennzeichnet. Bei einer Übersetzung durch das System würde das Objekt als solches übersehen. Im Gesamtprojekt kann man sich also nicht darauf verlassen, dass Objekte immer als Eigenname annotiert sind. Die zweite Besonderheit, die gleichzeitig ein Problem darstellen könnte, ist die Annotation von *left hand side*, das ein zusammengesetztes Substantiv und gleichzeitig eine Ortsangabe darstellt. Die Phrase *on the left hand side* ist gleichbedeutend mit *on the left*, was eine der bereits erwähnten Mehrdeutigkeiten der natürlichen Sprache darstellt. In der Tat wird *left hand side* hier als Adjektiv und zwei Substantive annotiert und nicht als einzelnes Substantiv. In einem solchen Fall könnte das System die Bedeutung der Phrase möglicherweise mit einer Sonderbehandlung für Richtungsangaben erkennen.

Ein Beispiel, bei welchem dem Markierer ein Fehler unterlaufen ist, ist der Satz *On the right in the foreground sits a Bunny on his hind legs facing southwest.*. Dieser Satz stammt aus Text 7. Die Annotation des Satzes sieht folgendermaßen aus:

> IN/On DT/the NN/right IN/in DT/the NN/foreground VBZ/sits DT/a NNP/Bunny IN/on PRP$/his IN/hind NNS/legs VBG/facing JJ/southwest ./.

Der Fehler befindet sich bei dem Wort *hind*. Dieses Wort steht in diesem Fall im Zusammenhang mit dem Wort *legs*, wobei *hind legs* auf Deutsch „Hinterläufe“ bedeutet. *Hind* ist also entweder zusammen mit *legs* ein zusammengesetztes Substantiv oder aber ein Adjektiv. Es wird hier allerdings vom Markierer als Präposition erkannt.

Eine weitere Unregelmäßigkeit findet sich bei der Annotation der Phrase *facing southwest*. Dabei wird *southwest*, wie oben zu sehen ist, als Adjektiv bezeichnet. Lässt man den Markierer den Satz *In the foreground on the left hand side there is a closed Mailbox facing southeast.* aus Text 1 annotieren, so erhält man folgendes Ergebnis:

> IN/In DT/the NN/foreground IN/on DT/the JJ/left NN/hand NN/side EX/there VBZ/is DT/a VBD/closed NNP/Mailbox VBG/facing

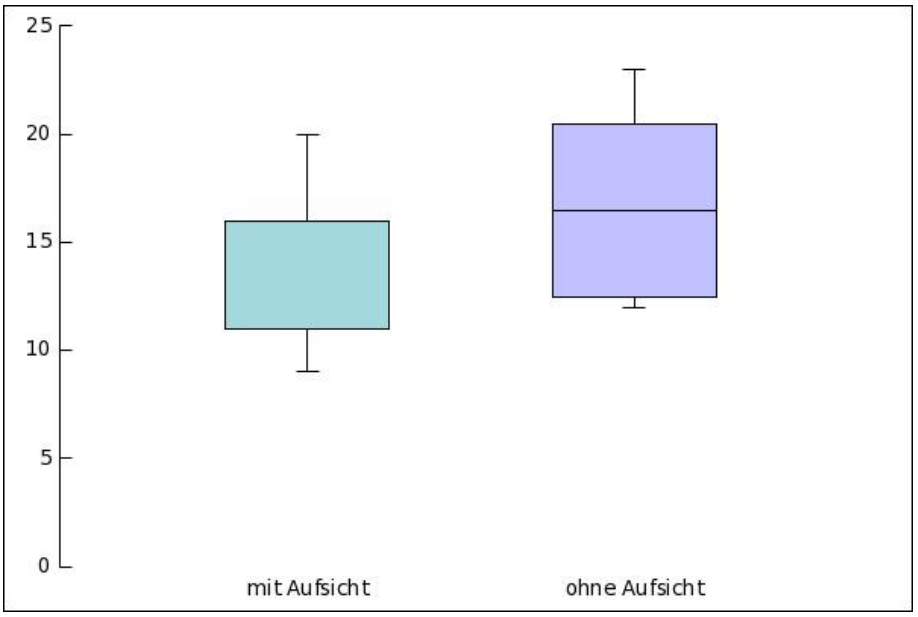

Abbildung 4.6.: Verteilung der Satzanzahl bei den beiden Probandengruppen

NN/southeast ./.

In der Phrase *facing southeast* sollte *southeast* der gleichen Wortart wie *southwest* angehören. Allerdings wird *southeast* hier als Substantiv annotiert.

Daher wurde ein zweiter Wortart-Markierer zur Überprüfung herangezogen, nämlich der von der Universität von Kopenhagen [CfS]. Dieser annotiert sowohl *hind* korrekt als Adjektiv, als auch beide Ortsangaben (*southeast* und *southwest*) als Adverbien (es handelt sich bei diesen Worten genauer um Lokaladverbien, was vom Wortart-Markierer jedoch nicht spezifiziert wird). Verwendet man einen dritten Markierer, den Stanford Core NLP [Sta], erhält man widerum ein anderes Ergebnis. Er klassifiziert zwar ebenfalls *hind* als Adjektiv, bezeichnet jedoch beide Ortsangaben als Substantive.

Nach dem heutigen Stand der Technik muss einkalkuliert werden, dass man bei der Annotation eines natürlichsprachlichen Textes ein fehlerhaftes Ergebnis erhält. Um den Einfluss dieses Problems zu vermindern, könnte man verschiedene Wortart-Markierer verwenden und deren Ergebnisse miteinander vergleichen.

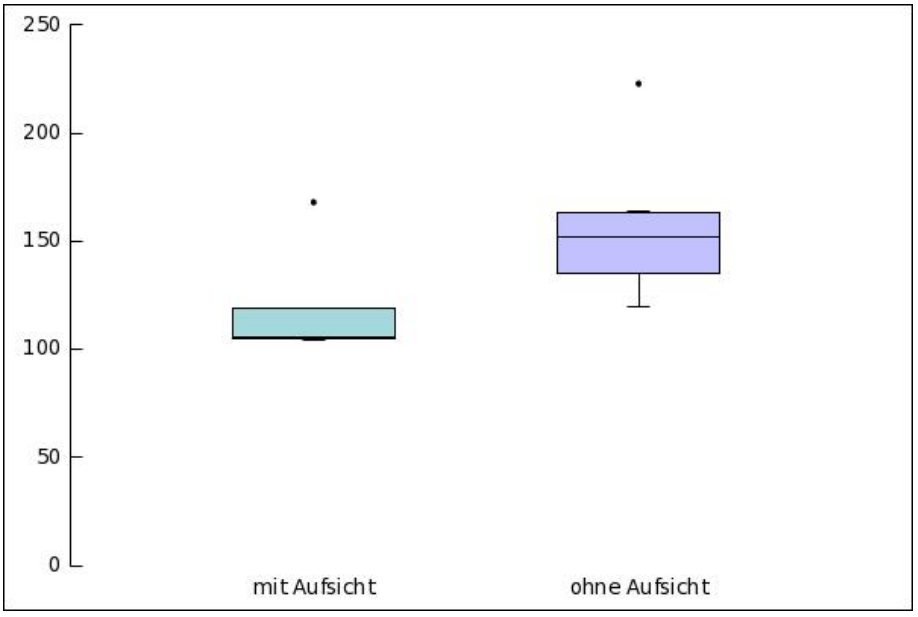

Abbildung 4.7.: Verteilung der Textlänge bei den beiden Probandengruppen

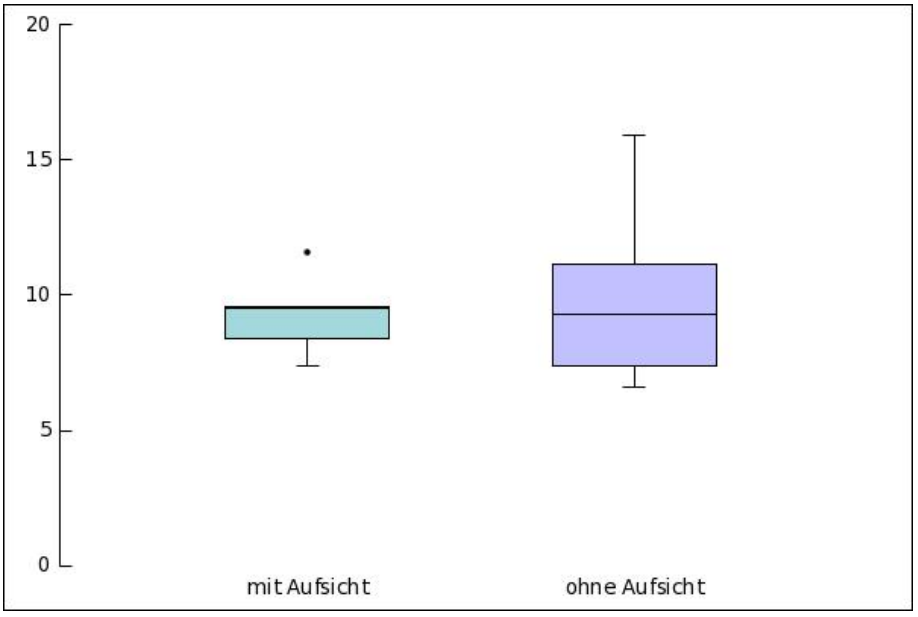

Abbildung 4.8.: Verteilung der durchschnittlichen Anzahl an Wörtern pro Satz

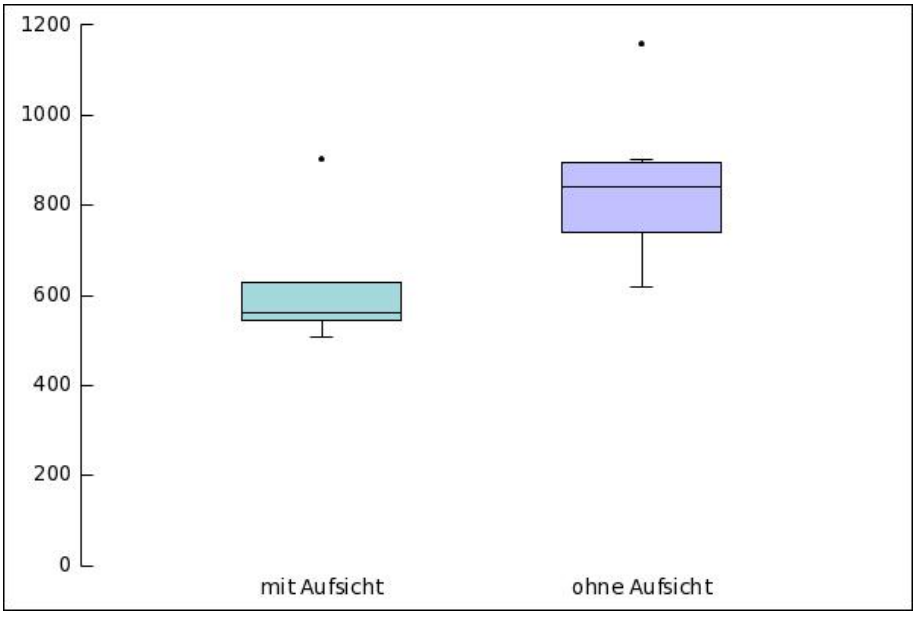

Abbildung 4.9.: Verteilung Anzahl an Zeichen

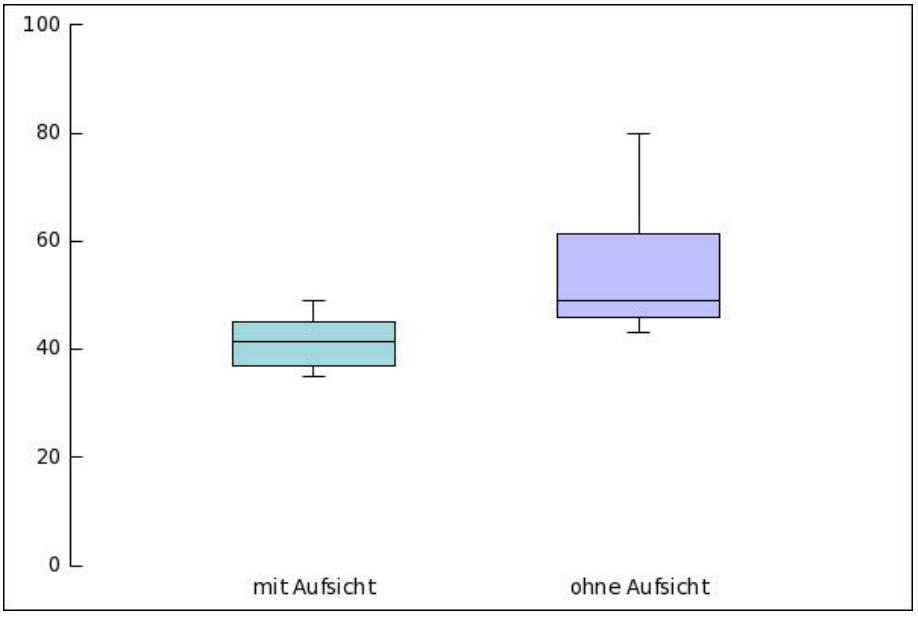

Abbildung 4.10.: Verteilung der Vokabulargröße

5. Zusammenfassung

Mit dem Sammeln und der Vereinheitlichung der Texte während der Durchführung der Studie konnte ein Textkorpus aus natürlichsprachlichen Texten aufgebaut werden. Diese Texte stammen alle aus der einzigen für unser Problem geeigneten Domäne, aus der Beschreibung einer Filmsequenz, die mit dem Programm Alice erstellt wurde. Es handelt sich also dabei um ein Alice-Korpus. Die wesentlichen Eigenschaften des Korpus sind in Tabelle 5.1 zusammengefasst. Da das Ziel des zu entwickelnden Systems darin besteht, Alice-Welten aus natürlichsprachlichen Beschreibungstexten aufzubauen, können die Texte im weiteren Projektverlauf verwendet werden. Sie ermöglichen die systematische Arbeit an einem Text- zu Alice-Übersetzer. Es wurden auch die Grundlagen, zum Beispiel in Form von Handlungsanweisungen, für eine eventuell notwendige Erweiterung des Korpus gelegt.

Außerdem wurden die im Korpus enthaltenen Texte analysiert und ihre Kennzahlen bestimmt. Dabei wurden zum Beispiel die Textlänge, die Satzlänge und die Vokabulargröße statistisch untersucht. Diese Merkmale sind bei der Entwicklung der Spracherkennungskomponente des Systems hilfreich.

Bei der Durchführung der Studie mit den Probanden war zu sehen, dass die Fragestellung nicht so einfach ist wie anfangs erwartet. Der Fragebogen, der als Anleitung erstellt wurde, enthielt unter anderem die Objekt- und Methodennamen. Es wurde festgestellt, dass sich selbst die Probanden mit Programmierkenntnissen, denen das Konzept von Objekten und Methoden geläufig ist, nicht an die Anweisungen hielten. Den Objekten wurden teilweise andere Namen gegeben und die Methoden mit eigenen Worten umschrieben, anstatt deren kürzere Namen aus dem Fragebogen zu entnehmen. Dies muss bei der weiteren Verarbeitung der Texte berücksichtigt werden.

Daraus kann man für zukünftige Studien lernen, den Fragebogen noch expliziter zu machen. Man sollte deutlicher auf die zu verwendenden Objekt- und Methodennamen hinweisen und

	Alice-Korpus
Sprache	Englisch
Medium	schriftlich
Beginn	2012
Umfang	1952 Wörter
Verwendung	Spracherkennung

Tabelle 5.1.: Eigenschaften des Alice-Korpus

diese gegebenenfalls noch mündlich erwähnen. Die Voraussetzung hierfür ist natürlich, dass man wünscht, dass die Probanden sich so genau an die Vorgaben halten und ihre Texte nicht freier formulieren. Es ist jedoch davon auszugehen, dass die Probanden auch dann nicht die Objekt- und Methodennamen verwenden, weshalb das System in der Lage sein muss, mit einer freieren Formulierung umzugehen.

Auch müssen Fragen, die bei der Bearbeitung des Fragebogens gestellt wurden, zukünftig berücksichtigt werden, so zum Beispiel die Frage nach dem Aussehen und der Farbe der Objekte. Für den Fall von Alice könnte man in den Fragebogen schreiben, dass das Aussehen fest vorgegeben ist und sich nicht verändern lässt. Außerdem wurde danach gefragt, ob bei den Methoden ein Parameter für die Häufigkeit der Ausführung angegeben werden muss. Man hätte also angeben können, welche Parameter bei jeder Methode jeweils übergeben werden, obwohl dies für natürliche Sprache nicht unbedingt üblich ist. Damit würde man sich allerdings weiter vom Ziel der einfachen Programmierung ohne sprachliche Einschränkungen entfernen.

Es wurde außerdem von wenigen Probanden geäußert, dass sie nicht gewusst hätten, was von ihnen erwartet wird, als sie den Beschreibungstext anfertigen sollten. Es hätte also genauer auf die Musterlösung hingewiesen werden können und diese womöglich noch einmal mündlich erklärt werden müssen. Dazu sollte in zukünftigen Fragebogen erwähnt werden, dass die Wortwahl frei ist (bis auf die vorgegebenen Objekt- und Methodennamen) und eigene Formulierungen durchaus erwünscht sind.

Bei der Analyse der Texte wurde festgestellt, dass Probanden ohne Programmierkenntnisse im Schnitt längere Sätze bilden. In ihren Texten liefern sie aber insgesamt weniger Informationen als Probanden mit Programmierkenntnissen. Obwohl das System für Personen ohne Programmierkenntnisse entwickelt werden soll, ist es anfangs einfacher, wenn eine Studie zur Anforderungserhebung mit Testpersonen stattfindet, die ein Ergebnis liefern, das sich gut weiterverarbeiten lässt. Da eine solche Studie den Aufbau eines Textkorpus zum Ziel hat, wäre eine Einseitigkeit der Probanden kein großes Problem. Die natürliche Sprache der Menschen mit Programmierkenntnissen unterscheidet sich nicht erheblich von der natürlichen Sprache anderer Menschen.

6. Ausblick

Da die Erstellung des Textkorpus nur einen Teil der Anforderungserhebung darstellt, könnte der nächste Schritt sein, die erhaltenen Texte mit geeigneten Werkzeugen systematisch zu annotieren. Danach können die Texte mit einer Alice-Ontologie verbunden werden, die Zusammenhänge zwischen Wörtern oder Satzteilen und Objekten, beziehungsweise Methoden herstellt. Mit dem Aufbau einer solchen Alice-Ontologie beschäftigt sich Oleg Peters in seiner Bachelorarbeit [Pet12].

Die Schwierigkeit bei der Annotation stellt, wie bereits erwähnt, die Vielfalt natürlicher Sprache und ihre mehrdeutigen Aussagen dar. Für dieses Problem gibt es jedoch, zumindest für die englische Sprache, mit den existierenden Werkzeugen schon sehr gute Lösungen. Hiermit beschäftigt sich die Studienarbeit von Sebastian Weigelt [Wei12].

Bei der Implementierung des Systems ist eine große Herausforderung, die Objekte in der virtuellen Welt zu platzieren. Da der Ursprung der Alice-Welt nicht ohne weiteres ersichtlich ist, es kein sichtbares Bezugssystem gibt und auch die Orientierung der Objekte mit natürlicher Sprache schwer zu beschreiben ist, ist es wahrscheinlich sinnvoll, die Objekte zunächst weiterhin per Drag-and-Drop zu platzieren. Ein anderer Ansatz wäre, mit einer gegebenen Szenerie anzufangen und zunächst die Handlungsbeschreibung umzusetzen.

Das System sollte, falls ein Benutzer eine existierende zusammengesetzte Methode beschreibt, diese erkennen können. Als Beispiel ist hier die Methode `monkey.pressButton()` zu nennen (siehe Anhang B). Innerhalb dieser Methode wird die Methode `lightBulb.switch()` aufgerufen. Eine explizite Erwähnung in der Beschreibung ist eigentlich nicht notwendig. Wird die Methode doch erwähnt, sollte das System in der Lage sein, dies zu erkennen und entsprechend zu handeln, damit die Methode nicht doppelt aufgerufen wird. Im umgekehrten Fall sollte das System auch in der Lage sein, falls eine Handlung vom Benutzer mehrfach mit den gleichen Worten beschrieben wird, diese als zusammengesetzte Methode zu definieren, auch wenn Alice diese noch nicht kennt.

Ein weiterer wichtiger Punkt, mit dem es sich zu beschäftigen gilt, ist die Implementierung eines komplexen Handlungsabläufes als Methode. Ist diese Methode in Alice nicht vorgegeben, müssten diese vom Spracherkenner so weit erkannt und interpretiert werden, dass eine Methode erstellt werden kann, die den gewünschten visuellen Effekt erzielt. Schreibt ein Benutzer beispielsweise, dass ein Kaninchen einen Brokkoli essen soll und in Alice ist die Methode `essen()` nicht definiert, so weiß das System in diesem Fall keine Lösung. Zunächst könnte das System den Benutzer in diesem Fall darum bitten, die Handlung genauer zu beschreiben. Soll diese Methode zusätzlich den Zustand eines anderen Objektes

verändern, wird diese Herausforderung noch größer. Das zu verändernde Objekt sollte dann gegebenenfalls mit angegeben werden. Ein wichtiger Punkt ist auch, dass die Parameter von natürlicher Sprache in Parameter mit passendem Typ übersetzt werden müssten.

Literaturverzeichnis

[Bub11] Noah Bubenhofer: *Einführung in die Korpuslinguistik: Praktische Grundlagen und Werkzeuge*. http://www.bubenhofer.com/korpuslinguistik/, 2011. Zugriff: 23.10.2012.

[Bur07] Lou Burnard: *Reference Guide for the British National Corpus (XML Edition)*. Research Technologies Service at Oxford University Computing Services, Oxford, 2007.

[CAB+00] Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, Kevin Christiansen, Rob Deline, Jim Durbin, Rich Gossweiler, Shuichi Kogi, Chris Long, Beth Mallory, Steve Miale, Kristen Monkaitis, James Patten, Jeffrey Pierce, Joe Schochet, David Staak, Brian Stearns, Richard Stoakley, Chris Sturgill, John Viega, Jeff White, George Williams und Randy Pausch: *Alice: Lessons Learned from Building a 3D System for Novices*. In: *CHI '00: Proceedings of the SIGCHI conference on Human factors in Computing Systems*, Seiten 486 – 493. ACM Press, 2000.

[CEE+01] Kai Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde und Hagen Langer: *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Spektrum, Heidelberg/Berlin, 2001.

[CfS] Kobenhavns Universitet Center for Sprogteknologi: *Brill's Tagger*. cst.dk/online/pos_ tagger/uk. Zugriff: 18.10.2012.

[dud] *Duden online*. www.duden.de. Zugriff: 24.10.2012.

[FK79] W. Nelson Francis und Henry Kucera: *Brown Corpus Manual*. Department of Linguistics, Brown University, Providence, Rhode Island, 1979.

[ice] *International Corpus of English*. ice-corpora.net/ice/. Zugriff: 23.10.2012.

[ill] *Cognitive Computation Group, University of Illinois at Urbana-Champaign*. cogcomp.cs.illinois.edu/demo/pos/. Zugriff: 18.10.2012.

[JAGL86] Stig Johansson, Eric Atwell, Roger Garside und Geoffrey Leech: *The Tagged LOB Corpus: Users' Manual*. Norwegian Computing Centre for the Humanities, Bergen, 1986.

[KL83] Lawrence Kasdan und George Lucas: *Star Wars: Episode VI - Die Rückkehr der Jedi-Ritter*, 1983.

[KM06] Roman Knöll und Mira Mezini: *Pegasus: First Steps toward a Naturalistic Programming Language*. In: *Proceedings of the 21st International Conference on Object-Oriented Programming, Systems, Languages and Applications*, Portland, Oregon, USA, oct 2006.

[Kuh10] Tobias Kuhn: *Controlled English for Knowledge Representation*. Dissertation, Universität Zürich – Wirtschaftswissenschaftliche Fakultät, 2010.

[LC06] Vinci Liu und James Curran: *Web Text Corpus for Natural Language Processing*. In: *EACL*, 2006.

[Lew85] Theodor Lewandowski: *Linguistisches Wörterbuch*. Quelle & Meyer, Heidelberg, 4. Auflage, 1985. Stichwort: „natürliche Sprache“.

[LZ06] Lothar Lemnitzer und Heike Zinsmeister: *Korpuslinguistik: Eine Einführung*. Narr, Tübingen, 2006.

[McE03] Tony McEnery: *Computational Linguistics*. In: Ruslan Mitkov (Herausgeber): *The Oxford Handbook of Computational Linguistics*, Oxford, 2003. Oxford University Press.

[MSM93] Mitchell P. Marcus, Beatrice Santorini und Mary Ann Marcinkiewicz: *Building a Large Annotated Corpus of English: The Penn Treebank*. Computational Linguistics, 19(2), 1993.

[Pet12] Oleg Peters: *Programmieren in natürlicher Sprache: Aufbau einer Alice-Ontologie*. Bachelor's Thesis, Karlsruher Institut für Technologie (KIT) – IPD Tichy, 2012. `http://www.ipd.kit.edu/Tichy/theses.php?id=202`.

[Sta] *Stanford CoreNLP*. nlp.stanford.edu:8080/corenlp. Zugriff: 18.10.2012.

[Teu98] Wolfgang Teubert: *Korpus und Neologie*. In: Wolfgang Teubert (Herausgeber): *Neologie und Korpus*, Tübingen, 1998. Narr.

[TKMS03] Kristina Toutanova, Dan Klein, Christopher Manning und Yoram Singer: *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*. In: *HLT-NAACL*, Seiten 252 – 259, 2003.

[Wei12] Sebastian Weigelt: *Programmieren in natürlicher Sprache: Aufbau einer Alice-Ontologie – Korpus-Ontologie-Assoziation –*. Diplomarbeit, Karlsruher Institut für Technologie (KIT) – IPD Tichy, 2012. `http://www.ipd.kit.edu/Tichy/theses.php?id=204`.

[Wol] *Wolfram Alpha*. www.wolframalpha.com. Zugriff: 18.10.2012.

Abbildungsverzeichnis

1.1. Die Entstehung einer Animation 2

2.1. Bildschirmfoto aus Alice 2.0 7

3.1. Aufbau der Welt „Penguin“ 12
3.2. Aufbau der Welt „Bunny“ 12

4.1. Beispiel für ein Boxplot-Diagramm 26
4.2. Verteilung der Satzanzahl bei den beiden Probandengruppen 27
4.3. Verteilung der durchschnittlichen Anzahl an Wörtern pro Satz 28
4.4. Verteilung der Anzahl an Zeichen 29
4.5. Verteilung der Vokabulargröße 30
4.6. Verteilung der Satzanzahl bei den beiden Probandengruppen 31
4.7. Verteilung der Textlänge bei den beiden Probandengruppen 32
4.8. Verteilung der durchschnittlichen Anzahl an Wörtern pro Satz 32
4.9. Verteilung Anzahl an Zeichen 33
4.10. Verteilung der Vokabulargröße 33

Tabellenverzeichnis

2.1. Etiketten und damit bezeichnete Wortarten des University of Illinois Wortart-Markierers [ill] 6
2.2. Kerneigenschaften der vorgestellten Korpora 8

3.1. Kerneigenschaften der Filmsequenzen 13
3.2. Benennung der Drehungen 17
3.3. Vergleich des Codes und der Beschreibung zur Sequenz „Penguin“ 18
3.4. Kerneigenschaften der Musterlösungen 18
3.5. Musterlösungen im Vergleich mit Anweisungen im Skript 19

4.1. Anzahl Sätze, Wörter und Zeichen in den Beschreibungstexten, sowie Merkmale der Probanden 25
4.2. Etiketten und damit bezeichnete Wortarten des University of Illinois Wortart-Markierers [ill] 29

5.1. Eigenschaften des Alice-Korpus 35

E.1. Eigenschaften der Verfasser der Beschreibungstexte 58

Anhang

A. Fragebogen für Probanden

Beschreibung einer Filmsequenz

In diesem Fragebogen geht es darum, eine kurze Filmsequenz in natürlicher Sprache zu beschreiben.

Es soll ein System entwickelt werden, das Anweisungen in natürlicher Sprache entgegennimmt und diese dann in Quellcode übersetzt, der vom Computer verstanden wird. Dies soll dazu dienen, Menschen mit geringen oder fehlenden Programmierkenntnissen die Möglichkeit zu bieten, selbst ein Programm zu entwickeln.

Für die Umsetzung dieses Projekts muss in einem ersten Schritt herausgefunden werden, wie Menschen solche Anweisungen an einen Computer formulieren würden. Zu diesem Zweck wurden mithilfe des Programmes Alice zwei Filmsequenzen entwickelt, von denen eine im Laufe dieses Fragebogens von Ihnen in natürlicher Sprache beschrieben werden soll.

Lesen Sie sich bitte die Anleitung zunächst vollständig durch, bevor Sie die Fragen beantworten.

Persönliche Informationen

1. **Name**: _____
2. **Alter**: _____
3. **Geschlecht:** □ männlich □ weiblich
4. **Programmierkenntnisse:**
   - □ Ich kann objektorientiert programmieren (z.B. Java, C++).
   - □ Ich kann prozedural programmieren (z.B. Pascal, Fortran).
   - □ Ich kann funktional programmieren (z.B. Haskell).
   - □ Ich habe keine Programmierkenntnisse.
   - □ Sonstiges: _____

Ein Beispiel

Starten Sie nun die Filmsequenz "penguin". Sehen Sie sich diese bis zum Ende an und lesen Sie anschließend die folgende Beschreibung, die Ihnen später bei der Anfertigung Ihrer eigenen Beschreibung als Richtlinie dienen soll:

> The ground is covered with grass. In the background there is a sunflower on the far right facing southwest. In the foreground there is a monkey in the middle facing southwest. To the monkey's right, there is a penguin facing south. To the monkey's left, there is a remote. Left of the remote, there is a light bulb. Behind the light bulb, to the right, there is a duck prince facing southeast. Behind the monkey, to the right, there is a bucket.
>
> The duck prince turns to face the monkey. The duck prince commands. The monkey sighs. The monkey turns to face the remote. The monkey jumps. The monkey presses a button. A very short time passes. The penguin turns to face the bucket. The penguin glides. The duck prince quickly turns to face the penguin. The duck prince scolds. A very short time passes. The monkey laughs. The monkey presses a button. The penguin jumps once.

Aufgabe

Starten Sie nun die Filmsequenz "bunny". Sehen Sie sich diese bis zum Ende an und beschreiben Sie sie anschließend bitte so genau wie möglich. Stellen Sie sich dabei vor, Sie wollten dem Computer Anweisungen geben, was er zu tun hat. Versuchen Sie bei der Beschreibung, die Reihenfolge der Handlung einzuhalten. Sollten Sie die Filmsequenz mehrmals betrachten oder pausieren wollen, so ist dies kein Problem.

Die Beschreibung soll bevorzugt in englischer Sprache verfasst werden, sollten Sie sich dazu nicht in der Lage fühlen, dürfen Sie auf Deutsch ausweichen.

Die folgenden Objekte werden in der Filmsequenz vorkommen und sollen in Ihrer Beschreibung mit dem angegebenen Namen angesprochen werden:

- Kaninchen (Name: Bunny)
- Frosch (Name: Frog)
- Briefkasten (Name: Mailbox)
- Palme (Name: PalmTree)
- Brokkoli (Name: Broccoli)

Es ist weiterhin zu beachten, dass die oben genannten Objekte die folgenden vorgegebenen Aktionen ausführen können:

Objekt	**Aktionen**
Bunny	eat, hop, open mailbox, tap foot
Frog	hop, ribbit
Mailbox	open door
alle Objekte	move, turn, roll, resize, say, think, play sound, move to, move toward, move away from, orient to, turn to face, point at, set point of view, set pose, stand up, move at speed, turn at speed, roll at speed, constrain to face, constrain to point at

Sollte es Ihnen nicht möglich sein, mit den vorgegebenen Aktionen eine im Film gesehene Handlung zu beschreiben, ist es vorzuziehen, diese mit anderen Worten zu beschreiben, als sie ganz wegzulassen.

5. Bitte beschreiben Sie die Filmsequenz.

6. **Ist Ihnen die Beschreibung der Filmsequenz schwer oder leicht gefallen?**
   sehr schwer □—□—□—□—□ sehr leicht
7. **Wie schätzen Sie die Länge Ihrer Beschreibung ein?**
   zu lang □—□—□—□—□ zu kurz

Vielen Dank, dass Sie sich für die Bearbeitung dieses Fragebogens Zeit genommen haben.

B. Alice-Skript für die Filmsequenz „Penguin“

```
world

Events

When the world starts
Do: world.my first method

Methods

world.my first method ( )
  No variables
    Wait 1 second
    DuckPrince turn to face monkey
    DuckPrince.command
    monkey.sigh
    monkey turn to face Remote
    monkey.jump
    monkey.pressButton
    Wait 0,5 seconds
    penguin turn to face bucket
    penguin.glide
    DuckPrince turn to face penguin duration = 0,2 seconds
    DuckPrince.scold
    Wait 0,5 seconds
    monkey.laugh
    monkey.pressButton
    penguin.jump times = 1


Duck Prince

Methods

DuckPrince.scold ( )
  No variables
    Do together
      DuckPrince.Chest.RightWing turn backward 0,2 revolutions
                        duration = 0,25 seconds
      Loop 3 times times
        Do in order
          DuckPrince.Chest.LeftWing turn left 0,25 revolutions
                        duration = 0,25 seconds
          DuckPrince.Chest.LeftWing turn right 0,25 revolutions
                        duration = 0,25 seconds
      Do in order
        DuckPrince roll left 0,1 revolutions duration = 0,25 seconds
        DuckPrince roll right 0,2 revolutions duration = 0,5 seconds
        DuckPrince roll left 0,2 revolutions duration = 0,5 seconds
        DuckPrince roll right 0,1 revolutions duration = 0,25 seconds
        DuckPrince.Chest.RightWing turn forward 0,2 revolutions
                        duration = 0,25 seconds
```

```
DuckPrince.command ( )
  No variables
    DuckPrince.Chest.LeftWing turn forward 0,2 revolutions
                          duration = 0,25 seconds
    DuckPrince.Chest.LeftWing turn backward 0,2 revolutions
                          duration = 0,25 seconds


penguin

Methods

penguin.jumping ( [123] height)
  No variables
    Do together
      penguin move up height meters duration = 0,5 seconds
      penguin.rightLeg.foot turn backward 0,1 revolutions
           duration = 0,5 seconds
      penguin.leftLeg.foot turn backward 0,1 revolutions
           duration = 0,5 seconds
    Do together
      penguin move down height meters duration = 0,5 seconds
           style = abruptly
      penguin.rightLeg.foot turn forward 0,1 revolutions
           duration = 0,5 seconds style = abruptly
      penguin.leftLeg.foot turn forward 0,1 revolutions
           duration = 0,5 seconds style = abruptly

penguin.glide ( )
  No variables
    penguin move up 0,5 meters duration = 0,25 seconds
    Do together
      penguin turn forward 0,25 revolutions duration = 0,5 seconds
      penguin.head turn backward 0,18 revolutions duration = 0,5 seconds
    penguin move forward 0.25 meters duration = 0,25 seconds
    If world.bucketStanding
      Do together
        Do in order
          Wait 0,18 seconds
          bucket.knockOver
        penguin move up 5 meters duration = 0,5 seconds style = end gently
        penguin.flapWings times = 2
    Else
      Do together
        penguin move up 3 meters duration = 0,5 seconds style = end gently
        penguin.flapWings times = 2
    Do together
      penguin turn backward 0,25 revolutions duration = 0,5 seconds
      penguin.head turn forward 0,18 revolutions duration = 0,5 seconds
    penguin move down 0,25 meters duration = 0,18 seconds
```

```
penguin.jump ( [123] times)
  No variables
    Loop times times time
      Do together
        penguin.flapWings times = 2
        penguin.jumping height = 0,5

penguin.flapWings ( [123] times)
  No variables
    Loop times times time
      Do together
        penguin.rightWing roll left 0.1 revolutions
             duration = 0,25 seconds
        penguin.leftWing roll right 0.1 revolutions
             duration = 0,25 seconds
      Do together
        penguin.rightWing roll right 0.1 revolutions
             duration = 0,25 seconds
        penguin.leftWing roll left 0.1 revolutions
             duration = 0,25 seconds


monkey

Methods

monkey.pressButton ( )
  No variables
    Do in order
      Do together
        monkey.right arm turn backward 0,1 revolutions
             duration = 0,5 seconds
        monkey.right arm.right forearm turn forward 0,1 revolutions
             duration = 0,5 seconds
        monkey.right arm.right forearm roll right 0,1 revolutions
             duration = 0,5 seconds
        monkey turn forward 0.03 revolutions duration = 0,5 seconds
      lightBulb.switch
      Do together
        monkey turn backward 0.03 revolutions duration = 0,5 seconds
        monkey.right arm.right forearm turn backward 0,1 revolutions
             duration = 0,5 seconds
        monkey.right arm.right forearm roll left 0,1 revolutions
             duration = 0,5 seconds
        monkey.right arm turn forward 0,1 revolutions
             duration = 0,5 seconds

monkey.sigh ( )
  No variables
    monkey.head turn backward 0,05 revolutions duration = 0,5 seconds
    monkey.head turn forward 0,08 revolutions duration = 0,5 seconds
    Do together
```

```
      monkey.head.right eyebrow move up 0,03 meters
           duration = 0,25 seconds
      monkey.head.left eyebrow move up 0,03 meters
           duration = 0,25 seconds
    Do together
      monkey.head.left eyebrow move down 0,03 meters
      monkey.head.right eyebrow move down 0,03 meters

monkey.laugh ( )
  No variables
    Do together
      monkey.head.right eyebrow move up 0,03 meters
           duration = 0,5 seconds
      monkey.head.left eyebrow move up 0,03 meters
           duration = 0,5 seconds
      monkey.head.jaw turn forward 0,05 revolutions
           duration = 0,25 seconds
    Do together
      Do in order
        monkey.head turn right 0,05 revolutions duration = 0,25 seconds
        monkey.head turn left 0,1 revolutions duration = 0,25 seconds
        monkey.head turn right 0,05 revolutions duration = 0,25 seconds
        monkey.head.jaw turn backward 0,05 revolutions
             duration = 0,25 seconds
      Do together
        monkey.head.left eyebrow move down 0,03 meters
        monkey.head.right eyebrow move down 0,03 meters

monkey.jump ( )
  No variables
    Do together
      Do in order
        Do together
          monkey move up 0,5 meters duration = 0,25 seconds
          monkey.left leg turn forward 0,1 revolutions
               duration = 0,25 seconds
          monkey.right leg turn forward 0,1 revolutions
               duration = 0,25 seconds
          monkey.right arm.right forearm turn forward 0,1 revolutions
               duration = 0,25 seconds
          monkey.left arm.left forearm turn forward 0,1 revolutions
               duration = 0,25 seconds
          monkey.left leg.left calf turn forward 0,1 revolutions
               duration = 0,25 seconds
          monkey.right leg.right calf turn forward 0,1 revolutions
               duration = 0,25 seconds
        Do together
          monkey move down 0,5 meters duration = 0,25 seconds
          monkey.left leg turn backward 0,1 revolutions
               duration = 0,25 seconds
          monkey.right leg turn backward 0,1 revolutions
               duration = 0,25 seconds
```

```
          monkey.right arm.right forearm turn backward 0,1 revolutions
               duration = 0,25 seconds
          monkey.left arm.left forearm turn backward 0,1 revolutions
               duration = 0,25 seconds
          monkey.right leg.right calf turn backward 0,1 revolutions
               duration = 0,25 seconds
          monkey.left leg.left calf turn backward 0,1 revolutions
               duration = 0,25 seconds
      monkey move forward 0,5 meters duration = 0,5 seconds


bucket

Methods

bucket.knockOver ( )
  No variables
    bucket move up 0,5 meters duration = 0,18 seconds
    bucket roll left 0,25 revolutions duration = 0,18 seconds
    bucket move left 0,5 meters duration = 0,18 seconds
    world.bucketStanding set value to false


lightBulb

Methods

lightBulb.switch ( )
  No variables
    If ( lightBulb . emissiveColor == (0, 0, 0) )
      lightBulb set emissiveColor to (1, 1, 0)
    Else
      lightBulb set emissiveColor to (0, 0, 0)
```

C. Alice-Skript für die Filmsequenz „Bunny“

```
world

Events

When the world starts
Do: world.my first method

Methods

world.my first method ( )
  No variables
    Wait 1 second
    bunny turn to face broccoli
    Loop 3 times times
      bunny.hop
    Wait 0,5 seconds
```

```
    bunny.eat food = broccoli
    Wait 0,5 seconds
    bunny turn to face frog
    Loop 2 times times
      bunny.tapfoot foot = left
    frog.ribbit
    frog turn left 0,25 revolutions
    Loop 3 times times
      frog.hop
    bunny turn to face mailbox
    Loop 3 times times
      bunny.hop
    bunny.openmailbox
    Do together
      bunny.upperBody.head turn forward 0.15 revolutions
           duration = 0,5 seconds
      frog turn to face bunny
    Loop 2 times times
      frog.hop
    frog.ribbit


bunny

Methods

bunny.eat ( [Obj] food)
  No variables
    Do together
      bunny turn forward 0.15 revolutions duration = 0,5 seconds
      bunny.rightLeg turn backward 0,15 revolutions duration = 0,5 seconds
      bunny.leftLeg turn backward 0,15 revolutions duration = 0,5 seconds
      bunny.upperBody.head turn forward 0,1 revolutions
           duration = 0,5 seconds
    food set isShowing to false
    Do together
      bunny turn backward 0,15 revolutions duration = 0,5 seconds
      bunny.leftLeg turn forward 0,15 revolutions duration = 0,5 seconds
      bunny.rightLeg turn forward 0,15 revolutions duration = 0,5 seconds
      bunny.upperBody.head turn backward 0,1 revolutions
           duration = 0,5 seconds

bunny.tapfoot ( [Direction] foot)
  No variables
    If ( foot == left )
      bunny.leftLeg.foot turn backward 0.1 revolutions
           duration = 0,25 seconds
      bunny.leftLeg.foot turn forward 0.1 revolutions
           duration = 0,25 seconds
    Else
      If ( foot == right )
        bunny.rightLeg.foot turn backward 0.1 revolutions
```

```
            duration = 0,25 seconds
        bunny.rightLeg.foot turn forward 0.1 revolutions
            duration = 0,25 seconds
      Else
        Do Nothing

bunny.openmailbox ( )
  No variables
    bunny.upperBody.leftArm turn backward 0,25 revolutions
         duration = 0,25 seconds
    mailbox.openDoor
    bunny.upperBody.leftArm turn forward 0,25 revolutions
         duration = 0,25 seconds

bunny.hop ( )
  No variables
    Do together
      Do in order
        Do together
          bunny move up 1 meter duration = 0,25 seconds
          bunny.rightLeg.foot turn forward 0,25 revolutions
                        duration = 0,25 seconds
          bunny.leftLeg.foot turn forward 0,25 revolutions
                        duration = 0,25 seconds
        Do together
          bunny move down 1 meter duration = 0,25 seconds
          bunny.leftLeg.foot turn backward 0,25 revolutions
                        duration = 0,25 seconds
          bunny.rightLeg.foot turn backward 0,25 revolutions
                        duration = 0,25 seconds
        bunny move forward 1 meter duration = 0,5 seconds


mailbox

Methods

mailbox.openDoor ( )
  No variables
    mailbox.door turn forward 0,5 revolutions duration = 0,5 seconds


frog

Methods

frog.ribbit ( )
  No variables
    Do together
      frog.head.jaw turn forward 0,05 revolutions
    frog.head.jaw turn backward 0,05 revolutions duration = 0,5 seconds
```

```
frog.hop ( )
  No variables
    Do together
      Do in order
        Do together
          frog move up 2 meters duration = 0,25 seconds
          frog.leftLeg turn forward 0,5 revolutions
               duration = 0,25 seconds
          frog.leftLeg.lowerLeg turn backward 0,25 revolutions
               duration = 0,25 seconds
          frog.rightLeg turn forward 0,5 revolutions
               duration = 0,25 seconds
          frog.rightLeg.lowerLeg turn backward 0,25 revolutions
               duration = 0,25 seconds
        Do together
          frog move down 2 meters duration = 0,25 seconds
          frog.leftLeg turn backward 0,5 revolutions
               duration = 0,25 seconds
          frog.leftLeg.lowerLeg turn forward 0,25 revolutions
               duration = 0,25 seconds
          frog.rightLeg turn backward 0,5 revolutions
               duration = 0,25 seconds
          frog.rightLeg.lowerLeg turn forward 0,25 revolutions
               duration = 0,25 seconds
        frog move forward 3 meters duration = 0,5 seconds
```

D. Musterlösungen

D.1. Musterlösung für die Filmsequenz "Penguin"

The ground is covered with grass. In the background there is a sunflower on the far right facing southwest. In the foreground there is a monkey in the middle facing southwest. To the monkey's right, there is a penguin facing south. To the monkey's left, there is a remote. Left of the remote, there is a light bulb. Behind the light bulb, to the right, there is a duck prince facing southeast. Behind the monkey, to the right, there is a bucket.

The duck prince turns to face the monkey. The duck prince commands. The monkey sighs. The monkey turns to face the remote. The monkey jumps. The monkey presses a button. A very short time passes. The penguin turns to face the bucket. The penguin glides. The duck prince quickly turns to face the penguin. The duck prince scolds. A very short time passes. The monkey laughs. The monkey presses a button. The penguin jumps once.

D.2. Musterlösung für die Filmsequenz "Bunny"

The ground is covered with grass. In the background there is a palm tree on the left. In the foreground there is a frog on the left facing east-southeast and a broccoli on the right. Behind the broccoli there is a bunny facing south-southwest. Behind the frog, to the left, there is a mailbox facing southeast.

The bunny turns to face the broccoli. The bunny hops three times. A very short time passes. The bunny eats the broccoli. A very short time passes. The bunny turns to face the frog. The bunny taps its left foot twice. The frog ribbits. The frog turns left by 90 degrees. The frog hops three times. The bunny turns to face the mailbox. The bunny hops three times. The bunny opens the mailbox. The bunny's head turns forward by a small amount while the frog turns to face the bunny at the same time. The frog hops twice. The frog ribbits.

Text	Geschlecht	Alter	Programmierkenntnisse	Anhang
1	m	26	objektorientiert	E.1
2	w	51	keine	E.2
3	m	52	keine	E.3
4	w	17	keine	E.4
5	m	19	keine	E.5
6	w	52	keine	E.6
7	m	75	keine	E.7
8	m	25	objektorientiert, prozedural, funktional	E.8
9	m	30	objektorientiert, funktional	E.9
10	m	31	objektorientiert	E.10
11	m	31	objektorientiert, prozedural, funktional	E.11
12	m	33	objektorientiert	E.12
13	m	24	objektorientiert, prozedural, funktional	E.13
14	m	31	objektorientiert, prozedural	E.14

Tabelle E.1.: Eigenschaften der Verfasser der Beschreibungstexte

E. Beschreibungstexte

Die Eigenschaften der Verfasser der Beschreibungstexte sind in Tabelle E.1 zusammengefasst.

E.1. Text 1

The ground is covered with grass, the sky is blue. In the background on the left hand side there is a PalmTree. In the foreground on the left hand side there is a closed Mailbox facing southeast. Right to the mailbox there is a Frog facing east. In the foreground on the right hand side there is a Bunny facing southwest. In front of the Bunny there is a Broccoli. The Bunny turns to face the Broccoli. The Bunny hops three times to the Broccoli. The Bunny eats the Broccoli. The Bunny turns to face the Frog. The Bunny taps his foot twice. The Frog ribbits. The Frog turns to face northeast. The frog hops three times to northeast. The Bunny turns to face the Mailbox. The Bunny hops three times to the Mailbox. The Bunny opens the Mailbox. The Bunny looks in the Mailbox and at the same time the Frog turns to face the Bunny. The Frog hops two times to the Bunny. The Frog ribbits.

E.2. Text 2

E.2.1. Originalfassung

Ich sehe eine Palme links im Bild, davor ein Briefkasten. Im Vordergrund sitzt ein Frosch links und ein Hase rechts im Bild. Der Hase hüpft und beugt sich nach vorne um etwas zu essen. Der Frosch betrachtet, was der Hase macht. Der Hase dreht sich zum Frosch und tapst mit der Hinterpfote. Der Frosch dreht sich mit dem Rücken zum Betrachter und hüpft nach hinten (in den Hintergrund). Der Hase hüpft zum Briefkasten und öffnet ihn. Er schaut hinein. Der Frosch dreht sich zurück zum Hasen und hüpft auch zum Briefkasten.

E.2.2. Englische Übersetzung

I see a palm tree on the left of the screen, a mailbox in front of it. In the foreground there sits a frog on the left and a hare on the right of the screen. The hare hops and bends forward to eat something. The frog looks at what the hare does. The hare turns to the frog and taps its hindpaw. The frog turns its back to the observer and hops to the back (into the background). The hare hops to the mailbox and opens it. He looks inside. The frog turns back to the hare and also hops to the mailbox.

E.3. Text 3

Ground is covered with green grass. Left background with palm. Left in front of palm: mailbox in brown american style. Left in front of mailbox: frog in green colour. Right front: bunny in white colour. Half of screen is blue sky. In front of bunny on right side: green broccoli. Bunny turns left and hops 3 times in direction of viewer of video. Bunny eats broccoli. Bunny turns right and taps two times with left foot. Frog says something to bunny. Frog hops 3 times from front left part of the scene to centre of the screen. Frog sits with back to viewer. Bunny turns a bit right to face the mailbox and hops 3 times to left part of the screen and sits down in front of the mailbox. Bunny opens mailbox with right hand. Bunny looks inside the mailbox. Frog turns and makes 2 hops back to mailbox. Frog sits behind bunny and looks into his direction. Frog says something to bunny.

E.4. Text 4

The floor is covered in grass. In the far right foreground is a Broccoli. In the far left is a Mailbox and in front of it is a Frog. Behind the Mailbox in the background is a PalmTree. Behind the Broccoli in the background is a Bunny. The Bunny is facing the south, the Frog is facing the Broccoli. The Bunny hops toward the Broccoli and eats it. The Bunny turns to face the Mailbox and taps its foot two times. The Frog ribbits and hops into the background. The Bunny hops toward the Mailbox and opens the Mailbox. The Frog hops toward the Bunny.

E.5. Text 5

The ground is covered with grass. In the background on the left there is a PalmTree. In the foreground in front of the PalmTree is a Mailbox. On the right side of the Mailbox there is a frog. On the right side in the foreground there is a Broccoli. Behind the Broccoli in the Background is a Bunny. The Bunny hops to the broccoli. The bunny eats the Broccoli and turns left to the frog. The bunny taps its foot. The frog ribbits. The frog turns to the Palm tree. The Frog hops in the background. The Bunny hops to the Mailbox. The Bunny opens the Mailbox. The frog turns to the Bunny. The frog hops to the Bunny.

E.6. Text 6

E.6.1. Originalfassung

The ground is covered with grass. The sky is blue. In the background there is on the left side a PalmTree. Links von der Palme ist ein Briefkasten. Auf der rechten Seite von der Palme sitzt ein Frosch. Links im Hintergrund ist Bunny. Vor Bunny ist Brokkoli. Bunny hoppelt mit 3 Sprüngen zu Brokkoli und frißt ihn auf. Dann dreht sich Bunny zu Frog nach links und winkt Frog mit der linken Pfote 2 mal zu. Frog öffnet das Maul und schließt es wieder. Frog dreht sich nach hinten und springt mit 3 Sprüngen nach hinten weg. Bunny hoppelt mit 3 Sprüngen zum Briefkasten, öffnet ihn und schaut hinein. Der Frosch dreht sich um und hüpft zurück zu Bunny und dem geöffneten Briefkasten.

E.6.2. Englische Übersetzung

The ground is covered with grass. The sky is blue. In the background there is a PalmTree on the left side. Left of the palm tree there is a mailbox. On the right side of the palm tree there sits a frog. In the background on the left there is Bunny. In front of Bunny there is Broccoli. Bunny hops with 3 leaps to Broccoli and eats it. Then, Bunny turns left to Frog and waves to Frog twice with the left paw. Frog opens the mouth and closes it again. Frog turns backward and hops away to the back with three leaps. Bunny hops to the mailbox with 3 leaps, opens it and looks inside. The frog turns around and hops back to Bunny and the open mailbox.

E.7. Text 7

On the upper half of the screen is a clear blue sky. On the lower half of the screen is a green meadow. On the right in the foreground sits a Bunny on his hind legs facing southwest. Before the Bunny stands a mushroom. On the left side in the foreground stands a palm with a brown trunk and green leaves. Before the palm and somewhat to the left of it stands a shut mailbox on a brown post. To the right of the mailbox a little more in the foreground sits a green frog with a yellow belly facing southeast in the direction of the mushroom. The Bunny turns his head and body slightly to the left facing exactly to the south. The Bunny jumps upward three times and then bends forward, lies down on the meadow and eats the mushroom. Then the Bunny stands up on his hind legs again and turns right facing west in the direction of the mailbox. The Bunny taps his left foot leg twice on the ground. Then the frog turns to the left facing northeast and making three jumps to the background. The Bunny hops to the mailbox, opens it and looks into it. The frog turns to the right now facing southwest and hops in two jumps to the right side of the Bunny.

E.8. Text 8

The ground is covered with grass. The sky is blue. In the background there is a palm tree on the left. In the foreground there is a broccoli on the right. Left to the broccoli there is a frog facing the broccoli. Behind the frog, to the left, there is a mail box facing southeast. Behind the Broccoli there is a rabbit facing south. The bunny turns to face the broccoli. The bunny hops towards the broccoli. The bunny eats the broccoli. The bunny turns to face the frog. The bunny taps its left foot two times. The frog ribbits. The frog turns to face north. The frog hops to the background. The bunny turns to the mail box. The bunny hops towards the mailbox. The bunny opens the mail box. The bunny turns its head down. Meanwhile the frog quickly turns to face the bunny. The frog hops towards the bunny. The frog ribbits. There is nothing happening for 3 seconds.

E.9. Text 9

The ground is covered with grass. The sky is blue. There is a palmtree on the left side in the background. On the right side there is a big white bunny facing southwest. In front of palmtree there is a mailbox. Right to the mailbox, there is a frog facing southeast. In front of the bunny, there is a broccoli. The bunny turns left to the south and hops three times to the broccoli. The bunny eats the broccoli. The bunny turns right to face the frog. The bunny taps two times on the ground. The frog ribbits. The frog turns left and hops three times to the northeast. The bunny hops three times to the mailbox. The bunny opens the mailbox door. The bunny looks inside the mailbox. The frog turns right to face the bunny. The frog hops two times back to the bunny. The frog ribbits.

E.10. Text 10

There is a bunny in the picture turning towards the camera. A piece of Brokkoli is in front of him. A frog sits left of the Brokkoli facing it. Behind the frog there is a mailbox. Behind the mailbox there is a palm. Everything is placed on grass. The bunny hops to the Brokkoli and eats it. Then it turns/moves to the rabbit and taps its foot. The frog turns left and hops 3 times away. The bunny then hops 3 times to the mailbox, opens it and looks down on the floor. While looking down, the frog turns around, hops to the bunny and ribbits.

E.11. Text 11

The ground is covered with grass. There is a palm on the left side of the ground, before the palm there is a mailbox. In front of the mailbox, there is a frog. On the right side of the ground, there is a broccoli. A rabbit is sitting behind the broccoli. The rabbit hops, moving toward the broccoli. The rabbit eats the broccoli. The rabbit turns to the frog and taps foot. The frog ribbits, turns left and hops to the back. The bunny hops to the mailbox, opens it, and looks inside the mailbox. Meanwhile, the frog turns to the bunny, moves forward, and ribbits.

E.12. Text 12

The ground is covered with grass. In the foreground to the right, there is a Broccoli. Behind the Broccoli, there is a Bunny facing south. In the foreground there is a Frog on the left facing southeast. Behind the frog, to the left, is a Mailbox. Behind the Mailbox, to the right, is a PalmTree. The Bunny hops to the Broccoli. It eats the Broccoli. The Bunny turns towards the frog. A very short time passes. The Bunny taps its foot. The Frog ribbits, turns to face north and hops away from the mailbox. A very short time passes. The Bunny hops to the Mailbox. The Bunny opens the mailbox. It is empty. The Bunny plays a sound (disappointment). A very short time passes. The frog turns around to face the bunny. The Frog hops towards the Bunny. A very short time passes. The Frog ribbits.

E.13. Text 13

The ground is covered with grass. In the background there is a palmtree on the far left. In front of the palmtree there is a mailbox. In the foreground to the right there is a broccoli. Right of the mailbox there is a frog facing the broccoli. On the right side behind the broccoli there is a bunny facing the frog. The bunny faces the broccoli and hops to it. Then the bunny eats the broccoli and turns to face the frog. The bunny taps foot twice. The frog ribbits, turns a half revelation to the left and hops away from mailbox. The bunny hops to the mailbox and opens mailbox. The frog turns to face the bunny, hops to the bunny and ribbits once.

E.14. Text 14

The floor is covered with grass. In the background on the left, there is a palm tree. In front of the palm tree, there is a postal mailbox facing southeast. In the foreground on the left, there sits a green frog with a yellow belly facing eastsoutheast. In the background on the right, there sits a white bunny facing southsouthwest. In south of the bunny, there is a broccoli. The bunny turns to the broccoli and hops 3 times towards it. The bunny bends down and eats the broccoli. The bunny sets pose, turns face to face to the frog and taps his left foot twice. The frog ribbits once, then turns north and hops 3 times. The bunny turns to the mailbox and hops 3 times. The bunny opens the mailbox's door with his left hand, lowers his head and looks into the mailbox. The frog turns around and hops twice towards the bunny.