Bachelor of Science

TextAnnotator-basierte Szenenerstellung mit Objekten aus ShapeNet

Carlo Hermanin de Reichenfeld

Abgabedatum: 11.11.2021

Text Technology Lab
Goethe-Universität Frankfurt am Main

Zusammenfassung

Ein aktuelles Forschungsthema ist die automatische Generierung von 3D-Szenen ausgehend von Beschreibungen in natürlicher Sprache. S.g. Text2Scene-Anwendungen sollen Objekte und räumliche Relationen in einer Texteingabe identifizieren und mit 3D-Modellen eine visuelle Repräsentation der Beschreibung konstruieren. Bisherige Ansätze kombinieren eine stichwortbasierte Erkennung von explizit gemachten Angaben mit vorher gelerntem Allgemeinwissen über die sinnvolle Anordnung von Objekten. Den Anwendungen fehlt jedoch ein tiefergehendes Verständnis von räumlicher Sprache.

Mit dem Annotationsschema ISOSpace können Texte mit detaillierten räumlichen Informationen angereichert und so für NLP-Anwendungen verständlicher gemacht werden. Bereits in einer früheren Arbeit wurde der SemAF-Annotator zum Erstellen von ISOSpace-Annotationen als Modul für den TextAnnotator entwickelt. In dieser Arbeit wurde der SemAF-Annotator zusätzlich um eine Funktionalität zur Szenenerstellung erweitert: Benutzer können einzelnen Wörtern in der Weboberfläche des TextAnnotators Objekte aus dem ShapeNet Datensatz zuordnen und diese in einer zweidimensionalen Darstellung einer Szene räumlich anordnen. Trotz einiger Einschränkungen durch die fehlende dritte Dimension lassen sich in vielen Fällen gute Ergebnisse erzielen. Die auf diese Weise erzeugten Szenen sollen später in Kombination mit den ISOSpace-Annotionen verwendet werden, um Text2Scene-Anwendungen zu entwickeln, die ein umfassenderes räumliches Verständnis aufweisen.

Kleinere Nebenaufgaben dieser Arbeit waren die Erweiterung des SemAF-Annotators um zusätzliche Annotationstypen sowie diverse Nachbesserungen der bereits bestehenden Funktionalität zur ISOSpace Annotation.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ein aktuelles Forschungsthema, das oft als Text2Scene bezeichnet wird, ist die automatische Generierung von 3D-Szenen basierend auf Beschreibungen in natürlicher Sprache (Chang, Savva und Manning 2014; Ma u. a. 2018). Eine erfolgreiche Umsetzung dieser Aufgabe würde es auch Benutzern, die keine Vorkenntnisse von meist sehr komplexer 3D CAD Software vorweisen können, ermöglichen, mit einfachen Texteingaben dreidimensionale Szenen zu erstellen. Denkbare konkrete Anwendungsfälle wären z.B. die Visualisierung von Unfallberichten und Tatortbeschreibungen, oder der Einsatz in den Bereichen der Pädagogik und Innenarchitektur.

Text2Scene-Anwendungen müssen erstens in der Lage sein, in einem Text genannte Objekte zu erkennen und geeignete 3D-Modelle zur Visualisierung auszuwählen. Zweitens müssen explizit beschriebene Relationen zwischen den Objekten bei der Anordnung in der Szene beachtet werden. Weiterhin enthalten räumliche Beschreibungen in natürlicher Sprache eine Vielzahl von impliziten Annahmen über die Lage der Objekte, die für Menschen offensichtlich sind, für Computeranwendungen aber eine Hürde darstellen. Wird beispielsweise „ein Raum mit zwei Regalen und einem Schreibtisch“ beschrieben, so ist davon auszugehen, dass die Regale an der Wand stehen und der Schreibtisch nicht direkt vor der Tür. Um Text2Scene-Anwendungen dieses Wissen beizubringen, wird eine große Menge von Trainingsdaten benötigt. Diese Arbeit befasst sich mit der Entwicklung eines Werkzeugs zum Erstellen solcher Daten.

Der TextAnnotator (Abrami, Mehler u. a. 2019) ist eine Webanwendung, mit der sich Annotationen verschiedener Art erzeugen lassen, welche anschließend zum Trainieren von NLP[1]-Anwendungen benutzt werden können. In einer früheren Arbeit wurde der SemAF-Annotator als Modul des TextAnnotators entwickelt, um die Annotation nach dem ISOSpace-Standard (Pustejovsky, Moszkowicz und Verhagen 2011; ISO 2020) zu ermöglichen. Dabei handelt es sich um ein standardisiertes Annotationsschema, das speziell für das Annotieren von räumlichen Beschreibungen entwickelt und als Teil des Semantic Annotation Frameworks (SemAF) (ISO 2016) veröffentlicht wurde. Neben der Kategorisierung von Wörtern je nach Art der enthaltenen räumlichen Information sieht ISOSpace die Annotation von Lagebeziehungen und Bewegungsabläufen vor. Da das Annotieren eine recht mühsame Aufgabe ist, wurde bei der Entwicklung des SemAF-Annotators ein besonderer Schwerpunkt auf die Benutzerfreundlichkeit gelegt. Die vor dieser Arbeit fertiggestellte Version des SemAF-Annotators wurde vom Autor im Rahmen eines Praktikums entwickelt und wird im Folgenden als Version 1.0 bezeichnet.

Die Hauptaufgabe dieser Arbeit war nun die Erweiterung des SemAF-Annotators um eine Funktionalität, die es dem Benutzer ermöglicht, eine visuelle Repräsentation (Szene) einer räumlichen Beschreibung zu erstellen. Zu diesem Zweck kann der Benutzer in der Weboberfläche des SemAF-Annotators einzelnen Wörtern Bilder aus dem ShapeNet Datensatz

[1]Natural Language Processing

(Chang, Funkhouser u. a. 2015) zuordnen, die die im Text genannten Objekte in der Szene darstellen sollen. Enthält ein Text z.B. die Beschreibung eines Wohnzimmers, so würde der Benutzer Bilder für die erwähnten Möbel aus dem ShapeNet Datensatz heraussuchen und diese in der Szene so anordnen wie im Text dargelegt. Die Szenen werden aus der Vogelperspektive in demselben Fenster angezeigt, in dem sich der zu annotierende Text befindet, so dass alle Informationen auf einen Blick sichtbar sind (Abbildung 1.1). Da die Szenen als Trainingsdaten für die automatische Generierung von dreidimensionalen Szenen dienen sollen, wird intern trotz der zweidimensionalen Benutzeroberfläche mit 3D-Koordinaten gearbeitet.

Kleinere Nebenaufgaben dieser Arbeit waren die Implementierung von weiteren Annotationstypen sowie Nachbesserungen der Funktionalitäten zur ISOSpace-Annotation von Version 1.0 des SemAF-Annotators.

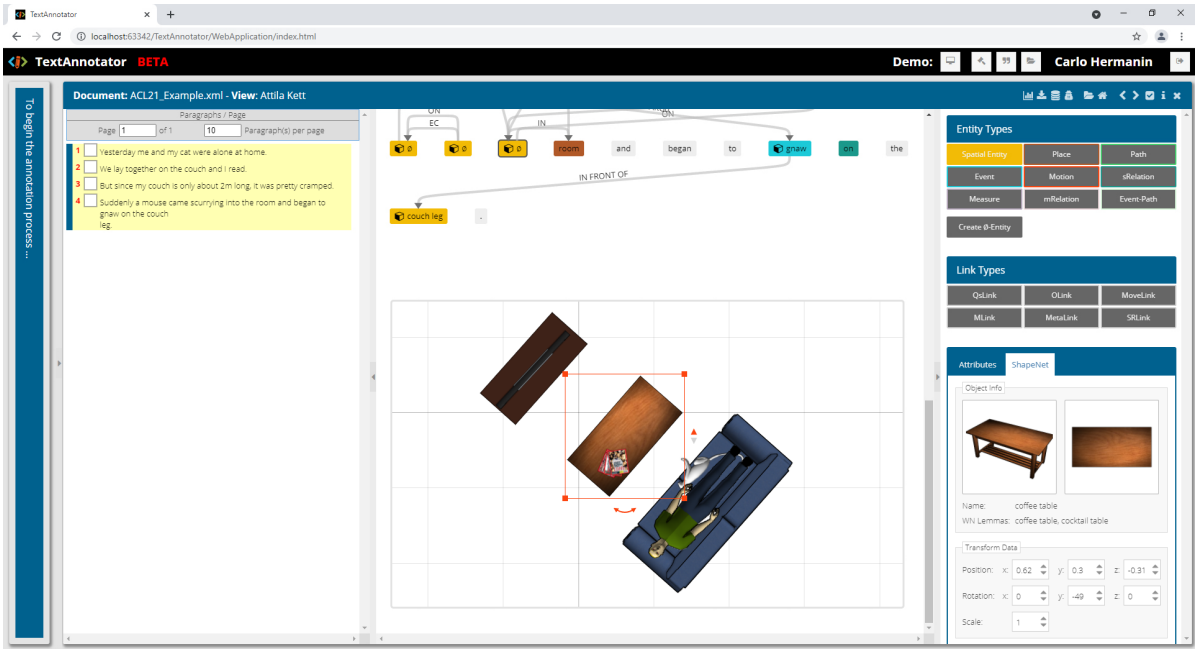


Abbildung 1.1: Szenenerstellung im SemAF-Annotator

2 Grundlagen

2.1 Semantic Annotation Framework

Das Semantic Annotation Framework (SemAF) (ISO 2016) ist ein nach ISO[1]-Kriterien entwickeltes Dokument, das mehrere standardisierte Schemata zur semantischen Annotation beinhaltet, darunter SemAF-Time zur Annotation von zeitlichen Informationen, SemAF-DA für Sprechakte, SemAF-SR für semantische Rollen und ISOSpace zur Annotation von räumlichen Beziehungen und Bewegungen. Für diese Arbeit sind die zwei letztgenannten Standards relevant. ISOSpace war bereits in Version 1.0 des SemAF-Annotators implementiert; der wesentlich simplere Standard SemAF-SR kam im Rahmen dieser Arbeit hinzu. In den nachfolgenden Abschnitten werden beide Standards kurz beschrieben.

ISOSpace

ISOSpace (Pustejovsky, Moszkowicz und Verhagen 2011; ISO 2020) ist ein Schema zur Annotation von verschiedenen Arten von räumlichen Informationen, die in natürlicher Sprache vorkommen können. Darunter fallen sowohl topologische Beziehungen und Richtungsbeziehungen zwischen Objekten als auch Objektmaße und Distanzen. Eine zentrale Bedeutung kommt zudem der Annotation von Bewegungen zu. Zu diesem Zweck beinhaltet ISOSpace sowohl s.g. Entitätstypen zur Markierung von einzelnen Wörtern oder Wortfolgen, als auch Relationstypen zur Annotation von Beziehungen zwischen im Text genannten Objekten.

Für jeden Typ sind eine Reihe von möglichen Attributen spezifiziert. Einige davon können optional ausgefüllt werden, um den Annotationsdaten noch genauere Informationen hinzuzufügen. Andere sind zwingend erforderlich, weil sie wesentliche Informationen über das entsprechende Typelement enthalten. Für Relationen *müssen* beispielsweise Referenzen auf die in Beziehung gesetzten Elemente sowie der genauere Untertyp der Relation in den Attributen eingetragen werden. Einige Entitätstypen müssen nicht zwangsläufig einem Wort zugeordnet werden, sondern können auch als s.g. leere Entitäten (engl.: *non-consuming tags*) erzeugt werden, die mit dem leere-Menge-Symbol (∅) dargestellt werden. Dadurch wird das Annotieren von räumlichen Informationen ermöglicht, die nur implizit im Text vorkommen. Es folgen zunächst eine kurze Erläuterung aller Typen und anschließend einige Annotationsbeispiele, die aus (ISO 2020) entnommen wurden.

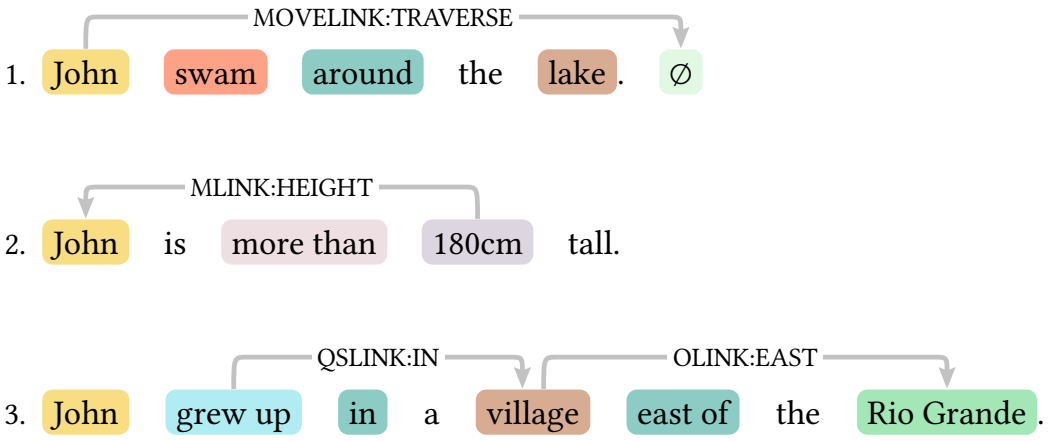
Enitätstypen: Mit den Typen Place, Path, und Spatial Entity können verschiedene Formen von räumlichen Entitäten markiert werden. Motion und Event sind zur Annotation von Ereignissen gedacht, die entweder durch eine Bewegung charakterisiert sind oder anderweitig für die räumlich Beschreibung relevant sind. Größen- und Distanzangaben werden mit dem Typ Measure markiert. Falls die Angabe eine mathematische Relation enthält,

[1] International Organization for Standardization

wird diese mit MRelation gekennzeichnet. Präpositionen, die auf eine räumliche Relation hinweisen, sollen den Typ SRelation zugewiesen bekommen. Der Typ Event-Path, der ausschließlich in Form von leeren Entitäten vorkommt, beschreibt den Pfad einer Bewegung, indem Start- und Endpunkte des Pfades in Attributen vermerkt werden.

Relationstypen: Der QSLink wird zum Annotieren von topologischen Beziehungen verwendet. Für Relationen muss im Attribut *rel_type* ein genauerer Untertyp eingetragen werden. Im Falle von QSLinks wird ein Wert aus dem RCC8+[1] ausgewählt. Richtungsbeziehungen werden mit OLinks gekennzeichnet. Kommt im Text eine Größenangabe vor, wird diese mittels eines MLinks (Measure-Link) mit der Entität verbunden, auf die sich die Angabe bezieht. MoveLinks verbinden eine sich bewegende Entität mit dem Event-Path, der den Pfad der Bewegung beschreibt.

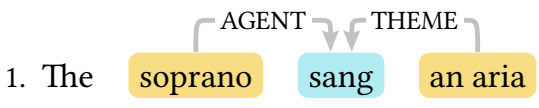
Beispiele:



SemAF-SR

Der Standard SemAF-SR (ISO 2014) befasst sich mit semantischen Rollen. Er enthält die Entitätstypen Eventuality zur Markierung von Eventualitäten (Ereignisse und Aktionen) und Entity zur Markierung von Entitäten, die an einer Eventualiät teilnehmen. Der Relationstyp SrLink ermöglicht die Verknüpfung von Eventualitäten mit den teilnehmenden Entitäten und die Annotation der semantischen Rollen, die die Entitäten dabei einnehmen.

Beispiel[2]:



[1]Der Region Connection Calculus: Neun verschiedene Beziehungen, die beschreiben, wie Objekte miteinander verbunden sind. S. z.B. `https://en.wikipedia.org/wiki/Region_connection_calculus`. Zusätzlich wird noch die Beziehung „in“ unterstützt.

[2]Aus (Ide und Pustejovsky 2017)

2.2 Koreferenzen

Ein wichtiger Aspekt von Natural Language Processing ist das Auflösen von Koreferenzen (engl.: coreference): In natürlicher Sprache kommt es häufig vor, dass eine Entität mehr als einmal genannt wird, wobei das referenzierende Wort oft nicht gleich bleibt, sondern durch ein Pronomen oder eine Nominalphrase ersetzt wird. In solchen Fällen sollen NLP-Anwendungen verstehen, dass alle Referenzen dieselbe physikalische Entität bezeichnen. Um diesen Zusammenhang annotieren zu können, wurden die Linktypen des SemAF-Annotators um einen s.g. MetaLink erweitert. Dieser kann zur Annotation von drei unterschiedliche Formen von Koreferenzen eingesetzt werden, die in den folgenden zwei Beispielen (aus Pustejovsky, Kordjamshidi u. a. 2015) erläutert sind.

1. Two cars are on the street. One of them turns left.

   Eine COREFERENCE liegt vor, wenn zwei Wörter dieselbe Entität bezeichnen.

   Eine SUBCOREFERENCE wird verwendet, um eine Teilmenge von Entitäten mit der Gesamtmenge zu verbinden.

2. John and Mary ∅ met at the store. They went shopping.

   Die SPLITCOREFERENCE kommt zum Einsatz, wenn durch einen Ausdruck mehrere Entitäten referenziert werden, die anderorts im Text einzeln genannt wurden. Die eingefügte leere Entität (∅) beschreibt die Gesamtmenge.

2.3 ShapeNet

ShapeNet (Chang, Funkhouser u. a. 2015) ist ein Datensatz von semantisch annotierten und kategorisierten 3D-Modellen. Von den insgesamt 12000 Modellen in 270 Kategorien machen Haushaltsgegenstände den größten Teil aus, aber auch einige natürliche Objekte wie Pflanzen und Tiere sind vorhanden. In dieser Arbeit werden nicht direkt die 3D-Modelle benutzt, sondern im Datensatz ebenfalls enthaltene 2D-Screenshots (*Thumbnails*) der Modelle, die die Objekte aus unterschiedlichen Perspektiven zeigen. Abbildung 2.1 zeigt die Thumbnails mit allen Perspektiven, die für jedes Modell zur Verfügung stehen.

Am Text Technology Lab[1] der Goethe-Universität Frankfurt ist ein ShapeNet-Webservice installiert, mit dessen Hilfe der Datensatz durchsucht werden kann. Der Webservice wird sowohl benutzt, um im Datensatz annotierte Informationen über die Objekte, wie den Namen, die Kategorie oder die Dimensionen der Objekte abzufragen, als auch um die Thumbnails herunterzuladen.

[1] `https://www.texttechnologylab.org/`

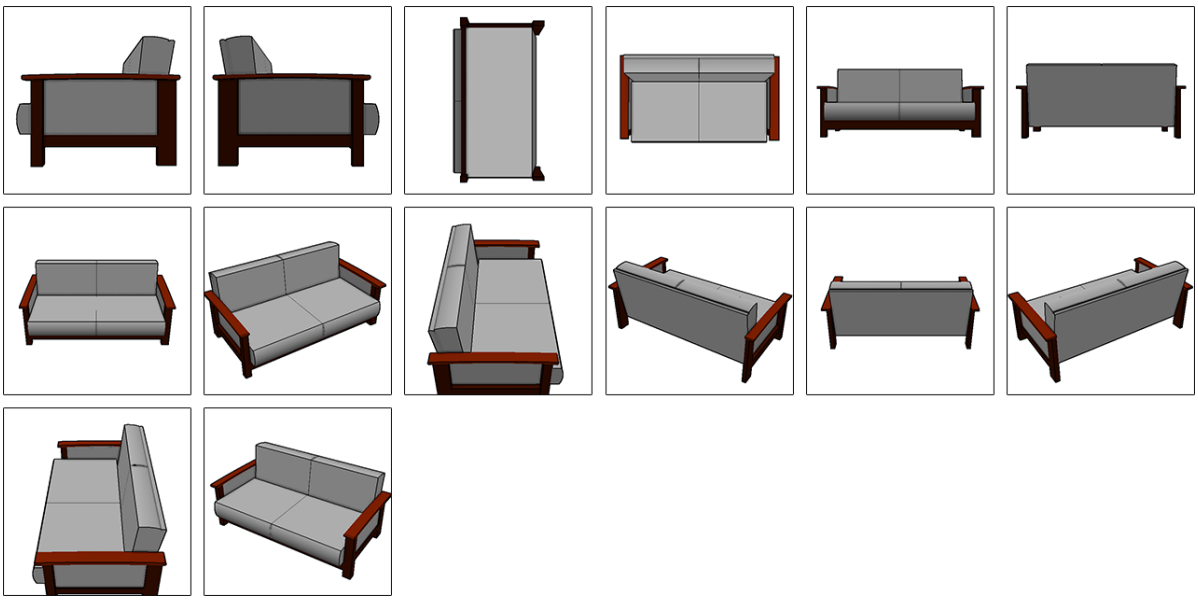


Abbildung 2.1: Die ShapeNet-Thumbnails

2.4 SVG

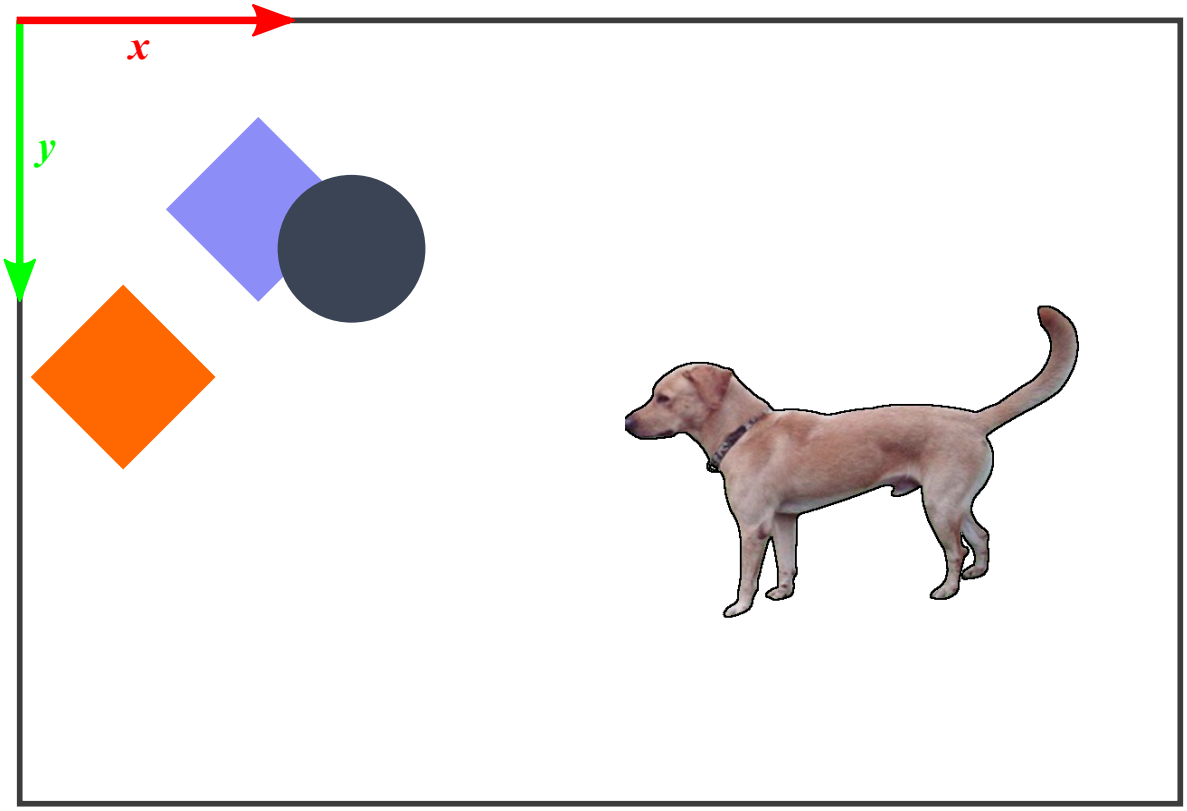
Der Begriff SVG steht für Scalable Vector Graphics. SVGs sind landläufig dafür bekannt, im Gegensatz zu den anderen im Web gängigen Formaten JPG, GIF und PNG auflösungsunabhängig und deswegen in allen Größen qualitativ hochwertig darstellbar zu sein. Weniger bekannt ist die Tatsache, dass es sich bei SVGs um nichts anderes als XML-Dateien handelt, so dass SVG-Bilder sich also mit jedem beliebigen Text-Editor erstellen lassen. Elemente und Formen können, ähnlich wie bei HTML, durch Tags definiert und durch Attribute näher spezifiziert werden. Abbildung 2.2 zeigt ein Beispiel mit dem dazugehörigen, etwas vereinfachten Code.

SVGs lassen sich nicht nur durch Referenzieren mittels des *<img>*-Tags in HTML-Dokumente einbinden, sondern auch direkt durch den *<svg>*-Tag in den HTML-Code eingliedern, sodass sie dann entweder von Hand direkt im Code, oder programmatisch mittels Javascript erstellt und editiert werden können. Alle SVG-Elemente sind, wie andere DOM[1]-Elemente, durch das *id*- oder *class*-Attribut mit Javascript anwählbar, sodass viele der Methoden zur DOM-Modifizierung auch auf SVG-Elemente angewendet werden können. Durch diese Eigenschaften eignen sich SVGs besonders gut zur interaktiven, visuellen Darstellung von Informationen und werden deshalb vom TextAnnotator benutzt, um Annotationen zu visualisieren.

Wie in Abbildung 2.2 sichtbar, ist der Ursprung des Koordinatensystems von SVG-Bildern in der linken oberen Ecke; die y-Achse zeigt nach unten. Eine für diese Arbeit essentielle Funktion der SVG-API ist die Möglichkeit, Elemente mittels des *transform*-Attributs räumlich

[1]DOM steht für Document Object Model und bezeichnet die objektbasierte Repräsentation eines HTML- oder XML-Dokuments. Durch die von dem DOM zur Verfügung gestellten Funktionen ist eine Interaktion mit den XML- oder HTML-Elementen mittels Javascript möglich.

zu transformieren. Diesem Attribut können die Operationen *translate*, *scale*, *rotate* und *matrix* zugeordnet werden, wobei die Reihenfolge, wie an den zwei Quadraten erkennbar, nicht immer beliebig ist[1]. Die Reihenfolge der Elemente im Code bestimmt, welches Objekt weiter oben gezeichnet wird. Bilder lassen sich mit dem *<image>*-Tag einbinden.




```svg
<svg>
    <rect id="rect-blue" fill="#8d8df7" width="23" height="23" x="0" y="0"
        transform="translate(45 20) rotate(45)" />
    <rect id="rect-orange" fill="#ff6700" width="23" height="23" x="0" y="0"
        transform="rotate(45) translate(45 20)" />
    <circle cx="61" cy="43" fill="#3a4454" r="13" />
    <image width="80" height="80" xlink:href="..."
        x="110" y="44" />
</svg>
```

Abbildung 2.2: Ein Beispiel-SVG und der dazugehörige Code

[1] In diesem Fall liegt das daran, dass um den Ursprung des SVGs rotiert wird.

2.5 UIMA

Bei UIMA (Unstructured Information Management Architecture) (Ferrucci u. a. 2009) handelt es sich um eine Architektur für Anwendungen, mit denen unstrukturierte Informationen analysiert werden sollen. Sie wird von zahlreichen NLP-Anwendungen, insbesondere auch vom TextAnnotator, implementiert. Ziel der UIMA ist eine einheitliche Repräsentation von Annotationen, unabhängig vom verwendeten Annotationsschema, um deren Austauschbarkeit und Wiederverwendbarkeit zu gewährleisten.

Zwei der wichtigsten Bestandteile der UIMA-Spezifikation sind die *Type System Descriptors (TSD)* zur Definition von Annotationschemata und die *Common Analysis Structure (CAS)* zur Speicherung der Annotationen. TSDs sind XML-Dateien, in denen mögliche Annotationstypen abstrakt definiert werden, die in den CAS-Dateien „instanziiert“ und einzelnen Textelementen zugeordnet werden können.

Abbildung 2.3 zeigt einen Ausschnitt aus dem Type System Descriptor für ISOSpace: Hier ist die Defintion der Typen Spatial Entity und Place mit jeweils einer ihrer Eigenschaften, *type* bzw. *country*, sichtbar. Neben primitiven Datenwerten (wie hier *String*) können Eigenschaften auch Referenzen auf andere Typelemente speichern. Weiterhin ist hier erkennbar, dass TSDs die Spezifikation von Typhierarchien erlauben: Der Typ Spatial Entity erbt vom Typ Entity und der Typ Place vom Typ Location. Beide Typen übernehmen somit alle Eigenschaften ihrer Obertypen.

Abbildung 2.4 zeigt einen Ausschnitt aus einem möglichen CAS-Dokument. Der zu annotierende Text wird getrennt von den erzeugten Annotationstypen im *<sofa>*-Tag (subject of analysis) gespeichert. Hier wurden die Typen Spatial Entity und Place jeweils einmal instanziiert und für die Eigenschaften type bzw. country wurden Werte eingetragen. Mittels der *begin*- und *end*-Eigenschaften verweisen die Typobjekte auf die entsprechenden Textstellen. Die instanziierten Typobjekte werden im Folgenden immer als *CAS-Elemente* bezeichnet.

Zusätzlich spezifiziert UIMA eine Reihe von Interfaces, die bei der Entwicklung von Komponenten, die Annotationsaufgaben erfüllen, implementiert werden müssen. Die Funktionalität jeder Komponente muss in einer XML-Datei, dem s.g. Component Descriptor, definiert werden. Durch Verweise auf diese Component Descriptors können neue Annotationsanwendungen im besten Fall bereits entwickelte Komponenten für die Erledigung von Teilaufgaben wiederverwenden. Anwendungen, die von mehr als einer Komponente Gebrauch machen, haben die Struktur einer Pipeline und werden als Aggregate Analysis Engine bezeichnet.

```
<typeDescription>
    <name>
        org.texttechnologylab.annotation.semaf.ISOSpace.SpatialEntity
    </name>
    <description/>
    <supertypeName>
        org.texttechnologylab.annotation.semaf.isobase.Entity
    </supertypeName>
    <features>
        <featureDescription>
            <name>spatial_entitiy_type</name>
            <description>Spatial Entitiy Type ( FAC | VEHICLE | PERSON
                | DYNAMIC_EVENT | ARTIFACT )
            </description>
            <rangeTypeName>uima.cas.String</rangeTypeName>
        </featureDescription>
        ...
    </features>
</typeDescription>
<typeDescription>
    <name>org.texttechnologylab.annotation.semaf.ISOSpace.Place</name>
    <description/>
    <supertypeName>
        org.texttechnologylab.annotation.semaf.ISOSpace.Location
    </supertypeName>
    <features>
        <featureDescription>
            <name>country</name>
            <description/>
            <rangeTypeName>uima.cas.String</rangeTypeName>
        </featureDescription>
        ...
    </features>
</typeDescription>
```

Abbildung 2.3: Ausschnitt eines Type System Descriptors für ISOSpace

```
<cas:Sofa xmi:id="1" sofaString="Carlo machte Urlaub in Rom."/>
<isospace:SpatialEntity xmi:id="2" sofa="1" begin="0" end="4"
    type="PERSON" />
<isospace:Place xmi:id="3" sofa="1" begin="23" end="25" country="Italy" />
```

Abbildung 2.4: Ausschnitt eines CAS-Dokuments

3 Ähnliche Arbeiten

3.1 Text2Scene

Die Entwicklung von Text2Scene-Anwendungen, die ein umfassendes Verständnis von räumlicher Sprache haben, ist die Motivation für diese Arbeit. In dem wegweisenden Projekt WordsEye (Coyne und Sproat 2001) wurden erstmals automatisch Szenen ausgehend von Beschreibungen in natürlicher Sprache generiert. Für Texteingaben wird zunächst eine semantische Repräsentation der enthaltenen räumlichen Informationen ermittelt, welche anschließend benutzt wird, um anhand von manuell erstellten Regeln (*depiction rules*) 3D-Modelle auszuwählen und zu positionieren. Einige Regeln für implizite räumliche Informationen sind ebenfalls bereits vorhanden. Aufgrund der beschränkten Anzahl manuell erstellter Regeln lässt sich dieser Ansatz schwer verallgemeinern.

Chang, Savva und Manning (2014) trainieren ihrem System räumliches Allgemeinwissen an, indem sie aus Szenen-Datensätzen A-priori-Wahrscheinlichkeiten für Objektrelationen lernen. Explizit in der Texteingabe gemachte Angaben werden wie bei Coyne und Sproat anhand von vorher festgelegten Stichwörtern erkannt. Das System ermöglicht eine nachträgliche Modifikation der Szenen durch den Benutzer und kann aus diesen Interaktionen lernen.

Ma u. a. (2018) legen eine Datenbank aus bereits erstellten Szenen an und suchen für Texteingaben eine passende Teilszene heraus, die die genannten Objekte enthält. Die Anordnung der Gegenstände ist somit von vornherein realistisch. Anschließend wird die Teilszene z. B. durch Hinzufügen von zusätzlichen Objekten oder Ändern von Relationen an die Texteingabe angepasst. Das System ist außerdem in der Lage, Szenen mit nicht explizit genannten Objekten anzureichern, sodass z. B. ein Schreibtisch automatisch mit Schreibutensilien ausgestattet wird. In einer Evaluation wurden die von Ma u. a. erstellten Szenen von Probanden als deutlich plausibler eingestuft als die von Chang, Savva und Manning.

Bislang wurden ISOSpace-Annotationen noch nicht zum Trainieren von Text2Scene-Anwendungen eingesetzt.

3.2 Annotationswerkzeuge

Es wurden bereits viele verschiedene Anwendungen zum Annotieren von Texten entwickelt. Da ISOSpace ein Annotationsschema mit speziellen Anforderungen ist, ist es aber nicht zu erwarten, dass nicht-spezialisierte Anwendungen die Annotation nach ISOSpace unterstützen. Beispielsweise ist WebAnno (Eckart de Castilho u. a. 2016) ein mächtiges webbasiertes Tool, das es dem Benutzer erlaubt, Annotationsschemata selbst zu definieren. Der angebotene Funktionsumfang ist für ISOSpace allerdings nicht ausreichend: Das Erstellen von leeren Entitäten ist nicht möglich und Relationen können immer nur für einen Entitätstyp definiert

werden, sodass es z.B. nicht möglich ist, einen Spatial-Entity-Token mit einem Place-Token zu verlinken.

Die Anzahl der Anwendungen, die ISOSpace unterstützen ist demnach sehr überschaulich: Neben dem SemAF-Annotator existieren nach bestem Wissen des Autors nur der VAnnotatoR (Spiekermann, Abrami und Mehler 2018) und MAE(2) (Stubbs 2011; Rim 2016). Diese Tools werden im Folgenden miteinander verglichen, wobei die Benutzerfreundlichkeit besondere Beachtung findet.

3.2.1 MAE und MAE2

MAE (Multi Annotation Enviroenment) (Stubbs 2011) ist eine Java-Anwendung zum Erstellen von Textannotationen, die 2011 veröffentlicht wurde. Die letzte Version erschien im Mai 2012 und wurde im Rahmen des SpaceEval-Tasks (Pustejovsky, Kordjamshidi u. a. 2015) zur Erstellung von ISOSpace-Annotationen benutzt. MAE hat zwar einen beachtlichen Funktionsumfang: Alle Funktionen, die für das Annotieren nach ISOSpace benötigt werden (Markieren von Entitäten, Erstellen von leeren Entitäten, Erstellen von Links, Eingabe von Attributen) sind vorhanden. Beim Arbeiten mit diesem Tool wird aber schnell deutlich, dass die Benutzerfreundlichkeit bei der Entwicklung nicht im Vordergrund stand.

Die Benutzeroberfläche ist vertikal in zwei Panels unterteilt: Im oberen Panel findet sich der zu annotierende Text, der im Unterschied zum TextAnnotator nicht vorverarbeitet ist, sodass er nicht in tokenisierter Form dargestellt wird. Das untere Panel zeigt Tabs für alle IsoSpace-Typen, in denen die erzeugten Elemente mit ihren Attributen tabellarisch dargestellt sind. Der Benutzer muss das Textelement, das er annotieren möchte, selbst mit der Maus markieren und kann nach anschließendem Rechtsklick in einem Kontextmenü den Annotationstyp auswählen. Besonders aufwändig ist die Eingabe von Referenz-Attributen: Die Id der Entität, die referenziert werden soll, muss zuerst durch Auswahl des entsprechenden Tabs nachgeschlagen und anschließend, nach Zurückwechseln in das vorherige Tab, manuell eingetragen werden. Leere Entitäten können über das Menü „NC elements“ erstellt werden, werden aber nicht im Text angezeigt. Das Erstellen von Links funktioniert recht komfortabel durch Strg+Mausklick auf die zwei Entitäten, die verknüpft werden sollen, und anschließender Auswahl des Linktyps in einem Popup-Fenster. Allerdings werden Links nicht visualisiert. Somit ist es verglichen mit dem SemAF-Annotator deutlich schwerer, zu erkennen, welche Entitäten mit welchen Linktypen verknüpft wurden.

Bei MAE2 (Rim 2016) handelt es sich um eine weiterentwickelte Version von MAE, die 2016 veröffentlicht wurde. Die aktuellste Version 2.2.11 erschien im April 2017. Das User-Interface und die Bedienung wurden nicht grundlegend verändert. Im Hinblick auf die Benutzerfreundlichkeit gibt es bei MAE2 kleine Fortschritte: Die Annotationstyp-Tabs im unteren Panel wurden mit farbigen Quadraten versehen, sodass auf einen Blick sichtbar ist, welche farblichen Textmarkierungen welchem Annotationstyp entsprechen. Wurde eine Entität oder ein Link erstellt, so wird das entsprechende Tab automatisch ausgewählt. Die sonstigen, oben beschriebenen Holprigkeiten bei der Bedienung blieben bestehen.

3.2.2 VAnnotatoR

Der VAnnotatoR (Spiekermann, Abrami und Mehler 2018) ist eine mit Unity3D entwickelte VR-Anwendung, die es dem Benutzer ermöglicht, in einer virtuellen 3D-Umgebung verschiedene Annotationsaufgaben mittels gestenbasierter Steuerung zu erledigen. Durch die dritte Dimension können mehrere Annotationsformen (Texte, Bilder, Webseiten) gleichzeitig in übersichtlicher Art und Weise dargestellt werden. Der TextAnnotator ist in den VAnnotatoR integriert, so dass innerhalb der virtuellen Umgebung auf dessen Funktionalität zugegriffen werden kann. Der VAnnotatoR kann verwendet werden, um Textelemente mit Objekten zu verknüpfen und so eine Szene zu erzeugen. In (Abrami, Henlein u. a. 2020) wurde der VAnnotatoR benutzt, um räumliche Beschreibungen mit *Hypertexten* zu annotieren. Dabei handelt es sich um eine Kombination aus einer Szene, die entsprechend der genannten Objekte und Relationen erstellt wurde, und vom Benutzer zusätzlich hinzugefügten Sätzen, die räumliche Informationen enthalten, die nur implizit in der ursprünglichen Beschreibung vorhanden sind. Die Hypertext-Annotationen sollen als Trainingsdaten für Text2Scene-Systeme dienen.

Im Vergleich zu der in dieser Arbeit entwickelten Funktionalität ist die Szenenerstellung im VAnnotatoR aufgrund der aufwändigeren Steuerung mit VR-Eingabegeräten vermutlich langsamer; da im Dreidimensionalen gearbeitet wird, lassen sich die Objekte allerdings genauer positionieren. Die Szenenerstellung im SemAF-Annotator soll daher in der Praxis synergistisch mit dem VAnnotatoR benutzt werden.

3.2.3 TextAnnotator

Der TextAnnotator (Abrami, Mehler u. a. 2019) ist ein Framework, das die UIMA-Architektur implementiert und die browserbasierte Erstellung und Visualisierung von Annotationen ermöglicht. Zum Zeitpunkt dieser Arbeit stellt der TextAnnotator acht verschiedene spezialisierte Werkzeuge zur Verfügung, mit denen verschiedene Formen von Annotationen erzeugt werden können.

Das Back-End des TextAnnotators wurde mit Java entwickelt und verwendet REST- und WebSocket APIs, um die Kommunikation mit dem Front-End, der Weboberfläche des TextAnnotators, zu ermöglichen. Der VAnnotatoR greift ebenfalls auf das TextAnnotator-Back-End zurück. Der *ResourceManager* (Gleim, Mehler und Ernst 2012) stellt die zu annotierenden Dokumente bereit und benutzt den *AuthorityManager* (Gleim, Mehler und Ernst 2012), um Benutzerrechte zu verwalten. Die unterstützen Annotationsschemata werden mit Hilfe von UIMA Type System Descriptors definiert und können daher leicht modifiziert, erweitert oder ausgetauscht werden. Die zu annotierenden Texte und die erzeugten Annotationen werden zusammen im UIMA-CAS-Format gespeichert und können als UIMA-xmi Dateien exportiert werden. Die Daten werden nicht in Dateien sondern mit Hilfe des *UIMA Database Interfaces* (Abrami und Mehler 2018) gespeichert. Dadurch ergeben sich die bekannten Vorteile von Datenbanken wie die Optimierung von Anfragen oder die Ermöglichung eines Mehrbenutzerbetriebs bei niedriger Redundanz. Soll ein neuer Text annotiert werden, der noch nicht im UIMA-CAS-Format vorliegt, so wird der *TextImager* (Hemati, Uslu und Mehler 2016) benutzt, um den Text zu konvertieren. Der TextImager wird auch für weitere Vorverarbeitungsaufga-

ben wie Tokensierung, Sentence Splitting, Lemmatisierung oder NER (Named Entity Recognition) verwendet.

Das Front-End des TextAnnotators ist in Javascript unter Verwendung des *ExtJS-Frameworks*[1] implementiert. Zur Darstellung der Annotationen werden SVG-Bilder verwendet, deren Inhalte mit Hilfe des Javascript-Frameworks *d3.js*[2] modifiziert werden. Durch den browserbasierten Ansatz können mehrere Benutzer dieselben Dateien bearbeiten und jeder Benutzer kann leicht von verschiedenen Geräten auf die Dokumente zugreifen.

3.2.4 SemAF-Annotator

Der SemAF-Annotator ist ein spezialisiertes Annotationswerkzeug, das als Modul des TextAnnotators entwickelt wurde, um das Annotieren nach dem ISOSpace-Standard zu ermöglichen. Version 1.0 des SemAF-Annotators ist der Ausgangspunkt für diese Arbeit. Im Folgenden wird die bereits implementierte Funktionalität beschrieben.

Designphilosophie

Die Menge der benötigten Trainingsdaten ist sehr groß. Das Annotieren ist jedoch eine recht mühsame, zeitintensive und repetitive Aufgabe. Ein Ziel bei der Entwicklung des SemAF-Annotators, sowie bei der Weiterentwicklung in dieser Arbeit, war es deshalb, ein hohes Maß an Benutzerfreundlichkeit zu erreichen. Der Annotationsprozess nach dem ISOSpace-Standard besteht im Wesentlichen aus nur drei Grundoperationen, die immer wieder ausgeführt werden müssen: 1) Markieren von Textelementen, 2) Erstellen von Relationen, 3) Eingabe von Attributwerten. Da jede dieser Operationen mehrere dutzend Mal pro Stunde von einem Annotator ausgeführt werden muss, lohnt es sich, die entsprechenden Anwendungsfunktionen derart zu optimieren, dass sie möglichst einfach, komfortabel, und schnell bedienbar sind. Das Ziel ist die Minimierung von unnötiger Frustration bei der Annotationsarbeit und die Steigerung der erstellbaren Annotationen.

Grundsätzlicher Aufbau

Der SemAF-Annotator benutzt das Model-View-Controller Entwurfsmuster, das für Webanwendungen, die mit ExtJS entwickelt werden, üblich ist. Die Klasse *SemAFPanelController* enthält die gesamte Anwendungslogik, die Klasse *SemAFPanelModel* die Definitionen der ISOSpace-Typen entsprechend des in UIMA TSDs vorgegebenen Schemas. Das Front-End verwaltet ein lokales Modell des zu annotierenden CAS-Dokuments; alle Änderungen am Dokument finden jedoch im Back-End statt. Allgemeinere Aspekte der Annotationsarbeit wie die Verwaltung von Dokumenten und Nutzerrechten, werden vom TextAnnotator gehandhabt und mussten daher nicht implementiert werden.

Der SemAF-Annotator extrahiert aus dem lokalen Modell alle für die Annotation von ISOSpace relevanten Informationen, d.h. alle ISOSpace-Typelemente sowie unmarkierte Textelemente, und kann durch eine vom TextAnnotator angebotene Bibliotheksklasse mit dem

[1] https://www.sencha.com/products/extjs/
[2] https://d3js.org/

Back-End kommunizieren. Nach jeder Benutzeraktion, z.B. dem Markieren eines Tokens, wird zuerst eine entsprechende Nachricht an das Back-End geschickt und erst nachdem eine Rückmeldung empfangen wurde, wird die Änderung in der Weboberfläche dargestellt. Will der Benutzer z.B. einen Token markieren, werden die benötigten Informationen, in diesem Fall die Textposition des Tokens und der gewünschte Typ der Markierung, an das Back-End geschickt. Dieses erzeugt daraufhin das entsprechende Typelement und schickt das modifizierte CAS-Dokument an das Front-End (*CAS-Update*).

Wie alle anderen Module des TextAnnotators ist der SemAF-Annotator bezüglich der erzeugten Änderung agnostisch, d.h. dass nicht nur die eine Änderung in der Benutzeroberfläche dargestellt wird, sondern, dass nach jedem CAS-Update das lokale Modell aus den empfangenen Serverdaten neu aufgebaut und nachfolgend neu in der GUI dargestellt wird. Da dieser Prozess mit d3.js implementiert wurde, müssen aber nicht alle GUI-Elemente neu gezeichnet werden. Vielmehr bietet diese Javascript-Bibliothek Funktionen an, die automatisch ermitteln, ob Elemente hinzugefügt, entfernt, oder verändert wurden, sodass nur diese Elemente neu gezeichnet werden müssen.

Aufbau der Benutzeroberfläche

Die Benutzeroberfläche des SemAF-Annotators ist in drei Panels unterteilt (Abbildung 1.1): Das linke Panel ist eine Komponente des TextAnnotators, die von den spezialisierten Werkzeugen, falls nötig, benutzt werden kann. Textsegmente, die zur Annotation angezeigt werden sollen, können durch Doppelklick, oder durch aufeinander folgende Klicks auf Anfang und Ende des gewünschten Segments ausgewählt werden. Alternativ lassen sich durch Klicken auf die blauen Rechtecke ganze Paragraphen auswählen. Im *Hauptpanel* in der Mitte werden ausgewählte Texte in tokenisierter Form dargestellt. Markierungen von Tokens werden farblich dargestellt, Relationen werden durch Pfeile visualisiert. Das rechte Panel (*Optionspanel*) bietet Buttons zum Erzeugen von Annotationen an und, falls ein Token oder Pfeil ausgewählt wurde, werden dessen Attribute in HTML-Formularlementen angezeigt und sind in diesen veränderbar.

Anwählen von Tokens

Bevor ein Token markiert werden kann, muss er vom Benutzer ausgewählt werden. Das ist entweder per Mausklick oder mit den Pfeiltasten der Tastatur möglich. Sobald ein Token ausgewählt wurde, wird er mit einer schwarzen Umrandung gekennzeichnet und die ISOSpace-Attribute des Tokens werden im Optionspanel angezeigt.

Markieren von Tokens

Durch Klicken eines der Buttons unter „Mark entity“ im Optionspanel können Tokens mit einer Typmarkierung versehen werden. Die Markierung lässt sich auch wieder verändern oder ganz entfernen. Wie bereits beschrieben werden Änderungen erst dargestellt, nachdem eine Nachricht an das Back-End gesendet und die Antwort empfangen und verarbeitet wurde. In der Praxis dauert dies wenige Millisekunden.

Zusammenfügen von Tokens

In einigen Fällen kann es notwendig sein Tokens zusammenzufügen: Z.B. bei Eigennamen, die aus mehreren Wörtern bestehen, wie „Frankfurt am Main“, würde man gerne die Unterteilung in einzelne Tokens aufheben und stattdessen den ganzen Namen zusammen markieren. Der TextAnnotator stellt dazu eine Funktion bereit, die von den Annotionsmodulen benutzt werden kann: Mit Drücken der Shift-Taste kann durch gleichzeitiges Ziehen der Maus von dem ersten zum letzten Token, die zusammengefügt werden sollen, ein einziger Token erstellt werden. Diese Benutzeraktion wird durch einen Pfeil, der am Punkt des Mausklicks startet und der Maus folgt, visualisiert.

Erstellen von leeren Entitäten

Kommt im Text eine räumliche Information vor, die sich nicht direkt auf einen Token bezieht, so ist es nach ISOSpace notwendig, eine leere Entität zu erstellen. Der SemAF-Annotator erlaubt dies durch Klicken auf einen der Buttons unter „Create element“ im Optionspanel. Die leere Entität wird, wie in der ISOSpace-Spezifikation, durch das leere-Menge Symbol (∅) dargestellt und hinter dem ausgewählten Token platziert. Anschließend wird das erzeugte Token automatisch ausgewählt, damit im Optionspanel das Attributpanel dargestellt und die Attributwerte vom Benutzer eingegeben werden können. Soll der Token wieder gelöscht werden, so ist dies entweder durch Drücken der Entf-Taste möglich, oder durch nochmaliges Klicken des Buttons, der zur Erzeugung der leeren Entität betätigt wurde.

Eingabe von Attributen

Wurde ein Token oder ein Relationspfeil vom Benutzer ausgewählt, so wird das entsprechende Attributpanel im Optionspanel angezeigt. Jeder Entitäts- und Relationstyp hat eigene Attribute, die der ISOSpace-Spezifikation entsprechen und in geeigneten Formularelementen angezeigt und verändert werden können. Um dem Benutzer unnötige Klicks zu ersparen, werden Attributänderungen automatisch an das Back-End geschickt, sobald das ausgewählte Element (Pfeil oder Token) abgewählt oder ein anderes ausgewählt wurde. Die Ausnahme sind Attribute, deren Änderungen sofort im Hauptpanel sichtbar sein sollten: Wird z.B. ein neuer Start- oder Endtoken einer Relation ausgewählt oder der Untertyp der Relation geändert, werden diese Änderungen sofort an das Back-End geschickt und nach der Rückmeldung im Hauptpanel dargestellt.

Entsprechend der ISOSpace-Spezifikation gibt es unterschiedliche Arten von Attributen: Bei einigen soll eine Texteingabe durch den Benutzer möglich sein, bei anderen eine Auswahl von vordefinierten Werten. Dies wird durch die aus Formularen im Internet bekannten HTML-Elemente, Textfelder und Komboboxen, ermöglicht.

Eine weitere Attributsart verlangt die Eingabe einer Referenz auf ein anderes Element. Ist in einem Text beispielsweise die geographische Höhe eines Ortes genannt, soll der entsprechende Token als Measure markiert werden und die Referenz im Attribut *elevation* des als Place markierten Tokens eingetragen werden. In diesem Fall muss der Benutzer die ID nicht von Hand eintragen, sondern kann, nach Klicken eines „Select“-Buttons den entsprechenden Token auswählen, welcher daraufhin im Attributfeld dargestellt wird und auch wieder

entfernbar ist. Der SemAF-Annotator zeigt dem Benutzer nach Klicken des Select-Buttons an, welche Typen eine gültige Auswahl darstellen, indem alle anderen Typen ausgeblendet werden. Im Beispiel oben würden also alle Tokens (und Pfeile), die nicht den Typ Measure haben ausgeblendet werden und wären dann nicht auswählbar.

Erstellen von Relationen

Es gibt zwei Möglichkeiten, Relationen zu erstellen. Die erste Möglichkeit ähnelt der Funktion zum Zusammenfügen von Tokens und ist eine weitere Funktion, die vom TextAnnotator implementiert ist und von den Annotationsmodulen bei Bedarf benutzt werden kann: Statt der Shift-Taste, drückt der Benutzer die Strg-Taste und zieht dann mit der Maus einen Pfeil von dem gewünschten Starttoken der Relation zum Endtoken. Der Pfeil, der dabei angezeigt wird, ist noch nicht der endgültige Relationspfeil, sondern dient nur der Visualisierung während der Auswahl der Tokens. Nachdem die Maustaste über dem Endtoken losgelassen wurde, erscheint der eigentliche Relationspfeil. Der auf diese Weise erstellte Pfeil hat noch keinen Typ. Dieser muss vom Benutzer im nächsten Schritt durch Klicken eines der Link-Buttons im Optionspanel unter „Create element“ ausgewählt werden. Um zu verhindern, dass Pfeile im Dokument bleiben, die keinen Typ haben und damit nicht Teil einer gültigen ISOSpace-Annotation sind, ist dem Benutzer keine andere Aktion erlaubt, bevor ein Typ ausgewählt wurde. Versucht der Benutzer den Pfeil ohne Typ abzuwählen, so wird dies unter Einblendung einer entsprechenden Fehlermeldung verboten.

Die andere Möglichkeit Relationspfeile zu zeichnen, funktioniert folgendermaßen: Der Benutzer wählt zuerst den gewünschten Starttoken der Relation aus. Nach Klicken eines der Link-Buttons erscheint ein Pfeil, der, da noch kein Endtoken ausgewählt wurde, ins Leere zeigt. Sobald der Benutzer anschließend mit der Maus über einen möglichen Endtoken fährt, wird ein Relationspfeil angezeigt, der die beiden Tokens verbindet. Nachdem der Endtoken vom Benutzer angeklickt wurde, wird der Relationspfeil endgültig eingezeichnet. Bei dieser Methode wird dem erzeugten Pfeil direkt ein Typ zugewiesen. Unabhängig davon, welche Methode zum Zeichnen eines Relationspfeils benutzt wurde, wird dieser nach dem Zeichnen automatisch ausgewählt, und das entsprechende Attributpanel angezeigt, sodass die Attribute sich sofort eingeben lassen. Alle Relationstypen haben Untertypen, die im Attributpanel ausgewählt oder eingegeben werden können und über den Linkpfeilen angezeigt werden.

Ist im Text ein Token vorhanden, der die Relation auslöst (s.g. *Trigger*), kann dieser im entsprechenden Attributfeld eingetragen werden. Bei dem Trigger-Attribut handelt es sich um ein Referenz-Attribut, dessen Eingabe wie oben beschrieben funktionert. Falls ein Trigger-Token ausgewählt wurde, wird dies durch einen gestrichelten Pfeil vom Token zum Relationspfeil dargestellt. Relationspfeile können, wie leere Entitäten, entweder durch Drücken der Entf-Taste oder durch nochmaliges Klicken des ausgewählten Link-Buttons, wieder entfernt werden.

Auswahl von ShapeNet Objekten

Im SemAF Annotator 1.0 war bereits eine Möglichkeit zur Auswahl von ShapeNet-Objekten für Tokens implementiert, die für die Zwecke dieser Arbeit erweitert wurde. In der bereits

implementierten Variante gibt es im Attributpanel der Tokens ein Textfeld, das den Namen des ausgewählten Objekts anzeigt und daneben einen „Select“-Button zur Auswahl des Objekts.

Nach Klicken des Buttons erscheint ein neues Fenster, das sowohl ein Textfeld zur Eingabe eines Suchstrings enthält, als auch eine Tabelle, in der die Ergebnisse der Suche, d.h. Informationen zu den einzelnen ShapeNet-Objekten, in Textform dargestellt werden. Sobald ein Eintrag vom Benutzer angeklickt wurde, schließt sich das Fenster und der Name des gewählten ShapeNet-Objekts wird in das Textfeld im Attributpanel übernommen. Die Thumbnails der Objekte werden in Version 1.0 weder im Auswahlfenster noch im Attributpanel angezeigt.

4 Realisierung

Vor Beginn der Realisierung wurden zwei Konzepte für die zu implementierende Funktionalität der Szenenerstellung entworfen, die im nachfolgenden Abschnitt dargelegt sind. Anschließend werden die einzelnen Aspekte der fertiggestellten Anwendung, beginnend mit der Benutzeroberfläche, beschrieben. Bei der Umsetzung wurde darauf geachtet, dass die erzeugten Szenen mit dem VAnnotatoR kompatibel sind.

4.1 Konzepte

Nach dem zuerst entwickelten Konzept sollte die Szenenerstellung in einem eigenen TextAnnotator-Tool implementiert werden. Später wurde ein zweites Konzept entworfen, welches letztlich umgesetzt wurde, das die Szenenerstellung direkt im SemAF-Annotator vorsieht, sodass der Benutzer nicht zwischen zwei Tools hin und her wechseln muss. Im Folgenden werden neben beiden Konzepten auch die Änderungen beschrieben, die während der Realisierung entstanden.

4.1.1 Umgesetztes Konzept

Der Ablauf der gewünschten Funktionalität lässt sich in vier Schritte unterteilen:

1. Der Benutzer wählt im Textpanel ein Textsegment aus, das durch eine Szene repräsentiert werden soll.
2. Der Benutzer klickt auf einen Button zum Erzeugen einer Szene. Die Szene wird erstellt und unterhalb des ausgewählten Textsegments angezeigt.
3. Der Benutzer weist einzelnen Tokens Grafiken aus dem ShapeNet Datensatz zu. Diese werden automatisch in der erzeugten Szene angezeigt.
4. Der Benutzer positioniert die Grafiken wie gewünscht in der Szene.

Es folgt ein detaillierteres Konzept für die einzelnen Schritte.

1. In Version 1.0 des SemAF-Annotators wird der gesamte Text des Dokuments in einem Textpanel auf der linken Seite in Paragraphen und Sätze unterteilt dargestellt. Durch Doppelklick auf einen Satz wird dieser zur Annotation ausgewählt; ein ganzer Paragraph kann durch Doppelklick auf das blaue Rechteck, das links neben den zu einem Paragraphen gehörigen Sätzen sichtbar ist, ausgewählt werden.

   Möchte ein Benutzer eine neue Szene erzeugen, so soll er zuerst das Textsegment, das durch die neue Szene repräsentiert werden soll, auswählen. Die neue Szene soll diesem Textsegment fest zugewiesen werden und im Textpanel soll der Name der Szene

als übergeordnetes Element, das die ausgewählten Sätze oder Paragraphen als Kindelemente enthält, angezeigt werden.

So wie in Version 1.0 mehrere Sätze durch Doppelklick auf ein Paragraph-Rechteck zur Annotation heranziehbar sind, sollen in der neuen Version auch alle zu einer Szene gehörigen Paragraphen oder Sätze durch Klicken auf den Namen der Szene auswählbar sein.

2. Unterhalb des ausgewählten Textes soll ein Button zum Erzeugen einer Szene erscheinen. Wird dieser Button vom Benutzer geklickt, so soll die Szene genau an dieser Stelle angezeigt werden.

3. In Version 1.0 des SemAF-Annotators war bereits eine Funktionlität zur Auswahl von ShapeNet-Objekten implementiert, die allerdings rein textbasiert abläuft und daher in dieser Arbeit erweitert werden soll. Dazu soll die Tabelle im Auswahlfenster durch ein Element ersetzt werden, dass die Thumbnails der Objekte, ähnlich wie z.B. bei einer Google-Image-Suche, anzeigt. Wurde ein Objekt ausgewählt, so soll nicht nur dessen Name im Attributpanel angezeigt werden, sondern auch das Thumbnail und möglicherweise noch weitere Informationen wie die zugehörigen WordNet Lemmata. Da die anderen ISOSpace Attribute dann aber im Attributpanel zu weit nach unten rutschen würden, bietet es sich an, alle ShapeNet Informationen inklusive Thumbnail in einem getrennten Tab darzustellen.

   Die Zuordnung eines ShapeNet-Objekts könnte durch ein kleines Icon neben dem Token im Hauptpanel visualisiert werden. Zusätzlich könnte dann bei Auswahl eines Tokens zwischen Mausklicks auf das Icon und Mausklicks auf den restlichen Bereich des Tokens unterschieden werden, sodass dann im Attributpanel automatisch entweder das ShapeNet-Tab oder das ISOSpace-Attribute-Tab ausgewählt werden könnte.

4. Wurden ShapeNet-Objekte für einzelne Tokens ausgewählt, sollen die entsprechenden Grafiken automatisch in der Szene, die wie in Schritt 2) beschrieben erstellt wurde, an einer gewissen Startposition positioniert werden. Der Benutzer soll die einzelnen Grafiken dann mit der Maus verschieben und transformieren können. Zur Transformation soll nach dem Anklicken, ähnlich wie bei CAD- oder Bildbearbeitungsanwendungen, eine Bounding-Box mit Bedienelementen zur Skalierung und Rotation um das Objekt herum gezeichnet werden.

4.1.2 Änderungen

Da die Dokumente, die mit Szenen annotiert werden sollen, so erstellt werden, dass sie immer nur die Beschreibung einer einzigen Szene enthalten, ist die Zuordnung einer Szene zu einem Textsegment wie in Schritt 1 beschrieben nicht nötig. Da es weiterhin keinen Sinn macht, Tokens ShapeNet-Objekte zuzuweisen, ohne diese in einer Szene anordnen zu wollen, fiel Schritt 2 ebenfalls weg. Die Darstellung der Szene erfolgt nun automatisch, sobald das erste Objekt ausgewählt wurde. Die automatische Auswahl des Attributpanel-Tabs je nachdem, welcher Bereich eines Tokens geklickt wurde, ist vielleicht eine nette Quality-of-Life-Funktion, die aus Zeitgründen aber nicht implementiert wurde.

4.1.3 Verworfenes Konzept

Nach diesem Konzept sollte die Funktionalität zum Erstellen einer Szene von der bereits bestehenden Funktionalität des SemAF-Annotators getrennt in einem eigenen Tool implementiert werden. Der Benutzer hätte zunächst in der Oberfläche des SemAF-Annotators die Auswahl und Zuweisung der Objekte zu den entsprechenden Tokens vornehmen und anschließend in das Tool zur Szenenerstellung wechseln sollen, um die Objekte dort anzuordnen.

Das Tool zur Szenenerstellung wäre nach diesem Konzept in zwei Bereiche unterteilt worden: Ein Panel auf der linken Seite, in dem alle ausgewählten ShapeNet-Objekte des Dokuments dargestellt wären, und der eigentliche Workspace zur Anordnung der Objekte auf der rechten Seite. Die Objekte hätten dann entweder durch Drag&Drop oder Doppelklick in der Szene platziert werden können.

Das Konzept wurde deshalb verworfen, weil das Hin-und-Her-Wechseln zwischen den beiden Tools in der Praxis für den Benutzer als zu verwirrend und unnötig kompliziert erachtet wurde. In diesem Konzept wird dem Benutzer durch das Objektpanel zwar ein Überblick über alle Objekte des Dokuments gewährt. Ob diese Funktion in der Praxis tatsächlich nützlich gewesen wäre, ist allerdings fraglich. Um so problematischer ist die Tatsache, dass der Benutzer im Tool zur Szenenerstellung keinen Überblick darüber hätte, in welchem Kontext die Objekte im Text genannt werden. Hier gab es zwar einige Überlegungen, diesem Problem Abhilfe zu schaffen: Relevante Textabschnitte hätten entweder in einem minimierbaren Fenster oder als Tooltip beim Drüberfahren mit der Maus über Objekte angezeigt werden können. Letztlich wurde aber aufgrund der Einschätzung, dass die zu erstellenden Szenen eher simpel ausfallen werden, das Auslagern der Funtionalität zu Szenenerstellung in ein eigenes Tool als nicht gerechtfertigt und in Anbetracht der oben beschriebenen Nachteile als nicht zweckmäßig beurteilt.

4.2 Benutzeroberfläche

Die Benutzeroberfläche wurde also entsprechend dem erstgenannten Konzept entwickelt. Eine Szene wird in einem Container-Element im Hauptpanel des SemAF-Annotators unter den Tokens des ausgewählten Textsegments angezeigt, allerdings nur dann, wenn das Textsegment mindestens ein ShapeNet-Objekt enthält. Ein Beispiel einer Szene ist in Abbildung 4.1 zu sehen. Abbildung 1.1 zeigt, wie die Szene in die restliche Benutzeroberfläche des SemAF-Annotators eingegliedert ist.

Da das Erstellen von Szenen eine Funktionalität ist, die üblicherweise in 3D-Editoren zu finden ist, orientiert sich auch das User Interface an den UIs dieser Programme: Der Boden der Szene wird durch ein Raster visualisiert. Durch Ziehen mit der Maus kann der Benutzer entweder einzelne Objekte oder das Raster, und damit die gesamte Szene, innerhalb des Container-Elements verschieben (Abbildung 4.2 links); mit dem Mausrad kann die Szene gezoomt werden (Abbildung 4.2 rechts). Unabhängig vom Zoomlevel kann auch ein größerer Ausschnitt der Szene angezeigt werden, indem das Hauptpanel des SemAF-Annotators vergrößert, oder das Text- oder Optionspanel ganz minimiert wird. Das Container-Element wird dann bei gleichbleibendem Seitenverhältnis entsprechend mitskaliert.

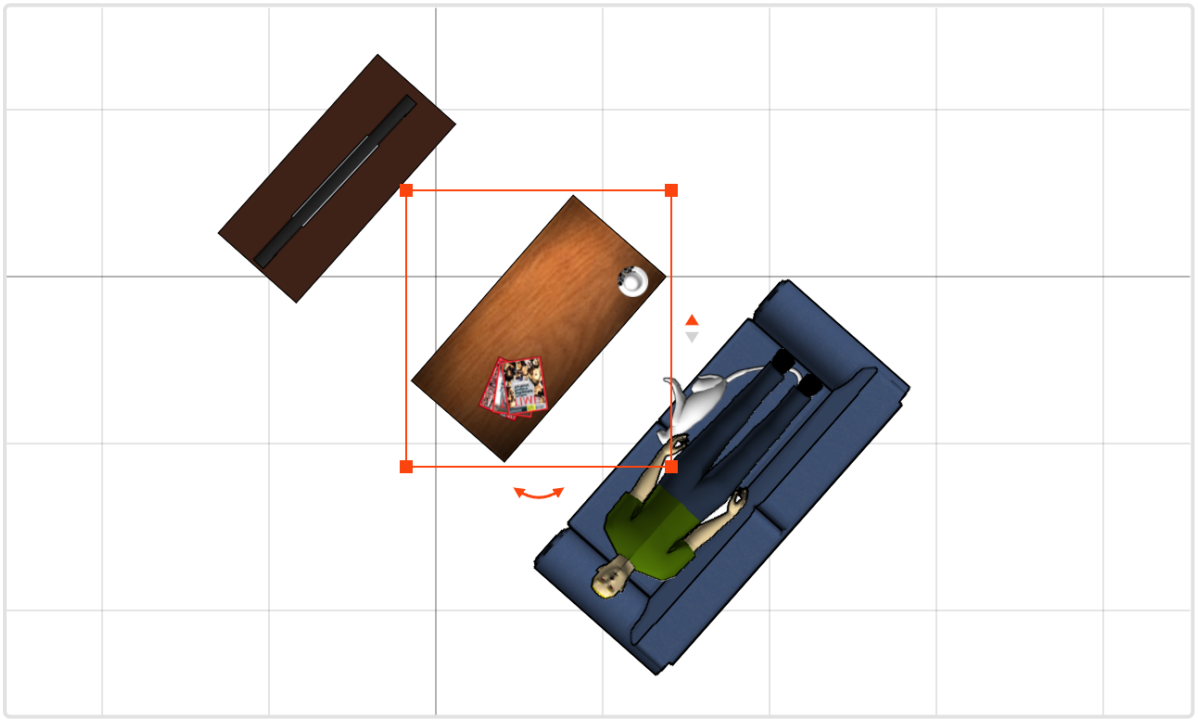


Abbildung 4.1: Die Szenenansicht

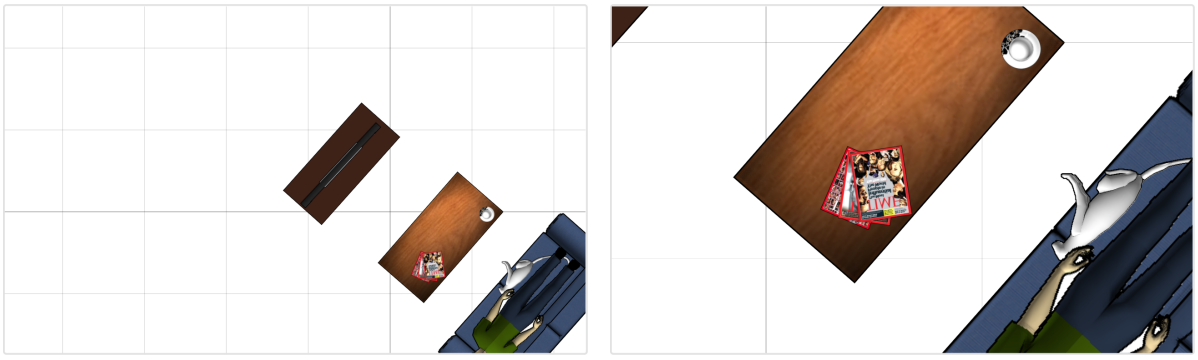


Abbildung 4.2: Verschieben und Zoomen einer Szene

Objekte werden entweder durch Bilder aus dem ShapeNet Datensatz (Thumbnails) (Abbildung 2.1) oder, falls kein geeignetes Modell enthalten ist, abstrakt durch geometrische Formen dargestellt. Ein Objekt lässt sich sowohl durch einen Klick auf das Objektbild in der Szene als auch durch einen Klick auf den entsprechenden Token auswählen. Umgekehrt wird beim Klick auf ein Objekt der entsprechende Token automatisch ausgewählt. Eine Bounding Box zeigt an, welches Objekt ausgewählt wurde und bietet die Möglichkeit, dieses zu skalieren, zu rotieren oder vertikal zu positionieren. Informationen über das ausgewählte Objekt werden im Attributpanel des SemAF-Annotators in einem eigenen Tab angezeigt.

4.2.1 Das Raster

Das Raster dient nicht nur der Darstellung des Bodens sondern hat auch die Funktion eines Koordinatensystems, das die internen Einheiten visualisiert, sodass jeder Pixel des Rasters internen Koordinaten entspricht und ein Objekt der Größe 1 genau ein Quadrat des Rasters ausfüllt. Die Koordinatenachsen sind im Vergleich zu den übrigen Linien etwas hervorgehoben.

Beim Zoomen wird das Raster nicht etwa einfach skaliert, sondern in jedem Zoomschritt mit entsprechend größeren oder kleineren Abständen neu gezeichnet. Diese Vorgehensweise ist notwendig, weil es bei dem einfachen Skalieren zu Darstellungsfehlern kommen würde[1]. Je nachdem, wie viel Platz beim Bearbeiten einer Szene benötigt wird, kann das Raster dynamisch erweitert oder auch wieder geschrumpft werden: Bewegt der Benutzer ein Objekt aus dem dargestellten Abschnitt heraus, oder transformiert er ein Objekt derart, dass nach der Transformation die Objektgrenzen aus dem Raster herausragen, dann wird das Raster bis über die neuen Objektgrenzen hinaus erweitert. Bei entgegengerichteten Benutzeraktionen wird das Raster wieder geschrumpft, wobei eine gewisse Initialgröße aber nicht unterschritten wird.

4.2.2 Attributpanel

Das Attributpanel des SemAF-Annotators wurde in ein *Ext-JS-Tabpanel* umgewandelt (Abbildung 4.3). Im ersten Tab finden sich, wie bereits in Abschnitt 3.2.4 beschrieben, die ISOSpace-Attribute des ausgewählten Tokens. Der Inhalt des zweiten Tabs hängt davon ab, ob dem Token bereits ein ShapeNet-Objekt zugewiesen wurde oder nicht. Falls nein, enthält dieses Tab nur zwei Buttons mit den selbsterklärenden Beschriftungen „add object“ und „add abstract“. Andernfalls werden der Name, die zugehörigen WordNet-Lemmata und zwei verschiedene Thumbnails des Objekts angezeigt. Das linke zeigt das Objekt aus einer 3D-Perspektive, was hilfreich sein kann, weil aus den achsenparallelen Perspektiven nicht immer klar erkennbar ist, um was für ein Objekt es sich handelt. Rechts wird das Thumbnail

[1]Bei eingeschaltetem Anti-Aliasing würden die Rasterlinien im herangezoomten Zustand verschwommen dargestellt werden; im herausgezoomten Zustand würden die Linien so fein gezeichnet werden, dass sie kaum sichtbar wären. Bei ausgeschaltetem Anti-Aliasing dagegen würden die Linien mit unterschiedlicher Dicke dargestellt werden (hohes Zoomlevel) oder teilweise ganz verschwinden (niedriges Zoomlevel). Somit bleibt das Neuzeichnen des Rasters als einzige Möglichkeit übrig, um eine ansprechende und fehlerfreie Darstellung zu gewährleisten. Bei den Bildern in diesem Dokument kann es je nach Zoomstufe ebenfalls zu Darstellungsfehlern kommen.

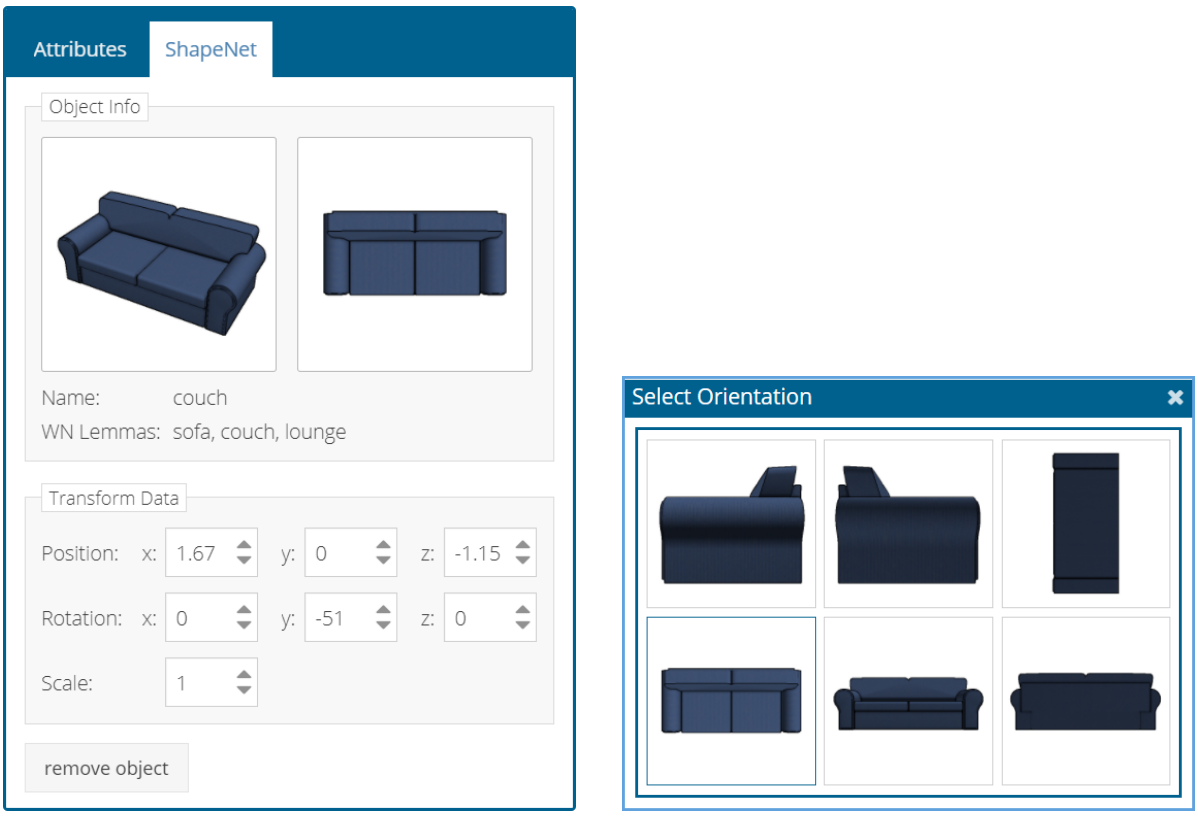


Abbildung 4.3: Das Attributpanel und das Fenster zur Auswahl der Objektrotation

angezeigt, das zur Darstellung in der Szene benutzt wird. Weiter unten finden sich einige Formularelemente, in denen die aktuelle Position, Rotation und Skalierung des Objekts angezeigt werden und verändert werden können. Alternativ lässt sich die Rotation auch durch direktes Auswählen des Thumbnails ändern. Das hierfür zuständige Fenster (Abbildung 4.3 rechts) erscheint nach einem Klick auf das rechte Thumbnail im Attributpanel.

4.2.3 Objektauswahl-Fenster

Klickt der Benutzer den „add-object“ Button im Attributpanel, erscheint ein Fenster (*ExtJS-Window*), in dem ein passendes Objekt für den ausgewählten Token ausgesucht werden kann (Abbildung 4.4). Dazu muss der ShapeNet Datensatz mit Hilfe des ShapeNet-Services durchsucht werden. Die erste Suchanfrage wird automatisch nach Öffnen des Fensters gestartet. Der Suchstring ist der Text des ausgewählten Tokens. Nach einer kurzen Ladeanimation erscheinen zunächst 20 Thumbnails, die relevante Objekte aus einer 3D-Perspektive zeigen. Der Benutzer kann weitere Suchergebnisse, falls vorhanden, anzeigen lassen, indem er in dem Fenster nach unten scrollt. Ab einer bestimmten Scrollposition werden die Thumbnails zu 20 weiteren Objekten nachgeladen[1]. Sollte die automatische Suche keine geeigneten Ergebnisse liefern, kann der Benutzer alternativ mit Hilfe eines Textfeldes nach einem anderen Begriff suchen.

[1]Eine Technik, die als *infinite scrolling* bekannt ist.

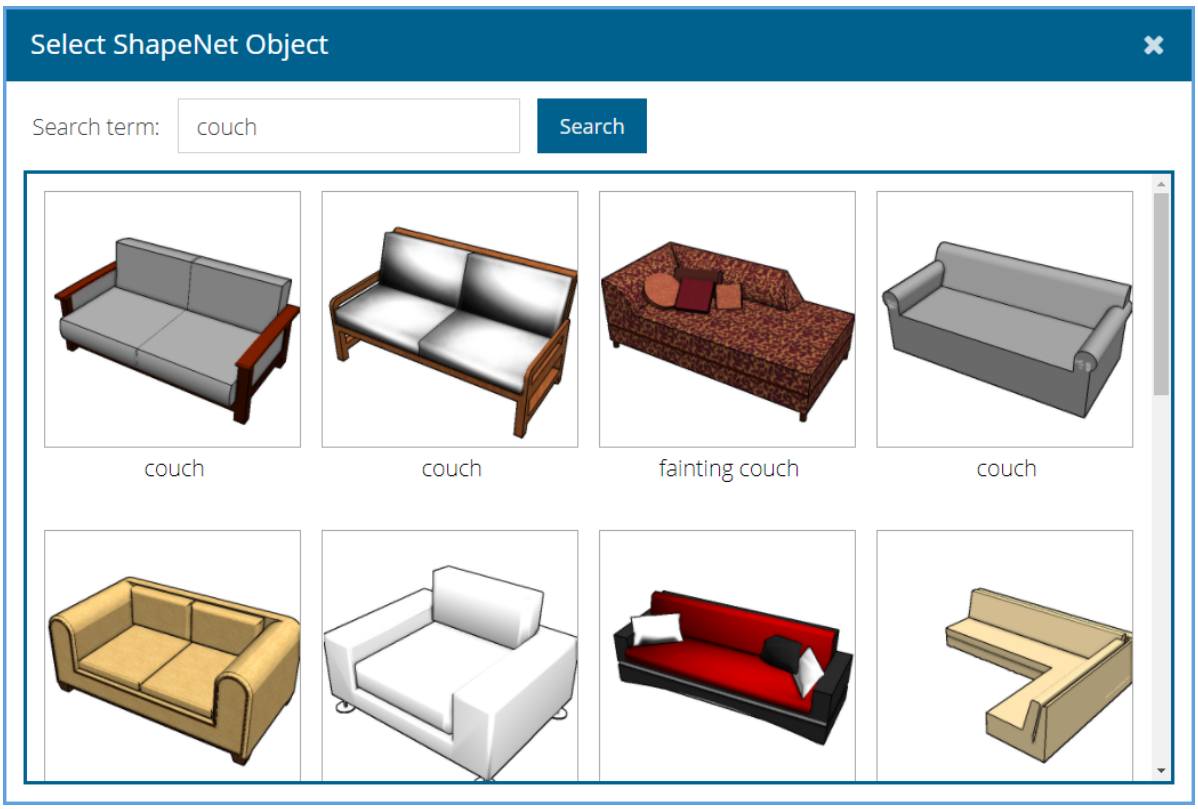


Abbildung 4.4: Das Fenster zur Auswahl eines Objekts

4.2.4 Abstrakte Objekte

Natürlich findet sich nicht für alle Objekte, die in einem Text vorkommen können, ein entsprechendes Modell im ShapeNet Datensatz. Sollen solche Objekte trotzdem Teil einer Szene sein, so kann die Darstellung im VAnnotatoR mit einer von fünf geometrischen Formen erfolgen, die für die Szenenerstellung im SemAF-Annotator übernommen wurden. Sie sind in Abbildung 4.5 zu sehen. Zur Auswahl stehen eine Fläche, ein Würfel, ein Punkt, ein Zylinder und eine Kugel. Diese abstrakten Objekte können in einem Fenster, das nach Klicken des Buttons „add abstract“ im Attributpanel erscheint, ausgewählt werden.

Abbildung 4.5: Die fünf verschiedenen abstrakten Objekte

4.3 Technische Details

4.3.1 Programmaufbau

Abbildung 4.6 zeigt alle entworfenen Klassen, die für die Szenenerstellung relevant sind[1]. Die Klasse *SemAFPanelController* ist für die gesamte Funktionalität, die zur Anzeige und Bearbeitung der ISOSpace-Annotationen benötigt wird, zuständig und benutzt vom TextAnnotator bereitgestellte Funktionen um mit dem Back-End zu kommunizieren. Der volle Funktionsumfang ist im Unterabschnitt 3.2.4 genauer beschrieben. Enthält ein Textsegment mindestens ein ShapeNet-Objekt, dann benutzt der SemAFPanelController den *SceneController* zur Darstellung der Szene.

Um eine möglichst lose Koppelung zwischen den beiden Funktionalitäten zu erreichen, ist die Steuerung sämtlicher UI-Elemente, die jenseits der Szene selbst für die Szenenerstellung benötigt werden, in der Klasse *SceneUIController* implementiert. Im Einzelnen geht es um das ShapeNet-Tab des Attributpanels, und die Fenster zur Auswahl eines ShapeNet- oder eines abstrakten Objekts. Neben den entsprechenden Layout-Definitionen enthält diese Klasse auch die Event-Handler-Methoden zur Verarbeitung der Benutzerinteraktionen mit diesen Elementen (im Diagramm nicht sichtbar, da es sich um anonyme Funktionen handelt, die direkt als Event-Handler der entsprechenden ExtJS-Klassen definiert wurden).

Der *SceneController* ist für das Zeichnen der Szene, die Ermittlung aller dafür benötigten Informationen und die Verarbeitung der Benutzerinteraktionen, d.h. Transformieren der Objekte, Verschieben oder Zoomen der Szene, zuständig. Das Zeichnen des Rasters ist in der Klasse *Grid* implementiert.

Der *ShapeNet-Helper* regelt die Kommunikation mit dem ShapeNet-Web-Service und speichert die erhaltenen Daten. Neben den Thumbnails werden für die Szenendarstellung auch weitere Informationen benötigt: Die Objektdimensionen zur Berechnung der richtigen Thumbnailgröße sowie der Objektname und die dazugehörigen WordNet Lemmata zur Darstellung im Attributpanel.

Da sowohl der Scene- als auch der SceneUIController auf die abstrakten Objekte zugreifen muss, sind diese in der Klasse *SemafAbstracts* in Form von Objektliteralen definiert. Um die Objekte in der Szene, im Attributpanel und im Auswahlfenster gleichermaßen zeichnen zu können, implementiert jedes Objekt eine *draw()*-Methode, die an den entsprechenden Programmstellen aufgerufen wird.

4.3.2 Allgemeiner Programmablauf

Nach dem Öffnen eines Dokuments wird im SemAFPanelController zunächst ein lokales Modell zur Darstellung der ISOSpace-Annotationen aus den vom Back-End erhaltenen Daten aufgebaut. Bei diesem Schritt werden die CAS-Elemente der aktuellen Textauswahl an den Scene- und den SceneUIController weitergeleitet. Enthält das Textsegment mindestens ein ShapeNet-Objekt, wird das Container-Element für die Szene unter dem tokenisierten Text hinzugefügt.

[1] Es ist nur eine Auswahl der wichtigsten Methoden und Attribute aufgeführt. *Object* und *Abstract* sind technisch gesehen nicht als Klassen sondern als Objektliterale definiert.

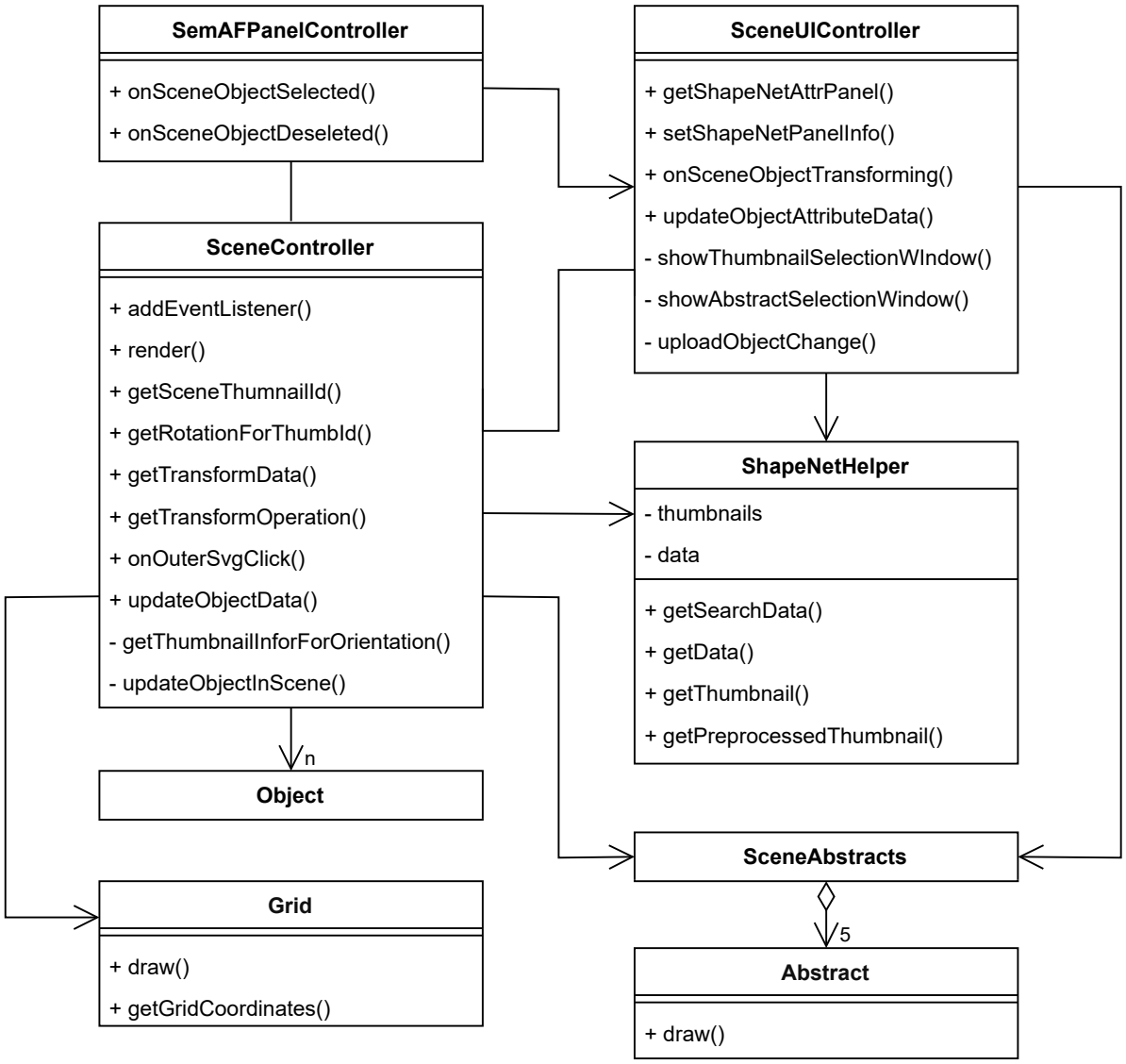


Abbildung 4.6: Die an der Szenenerstellung beteiligten Klassen

Der SceneController baut die grundsätzliche Struktur des Szenen-SVGs auf, zeichnet das Raster und extrahiert aus den CAS-Elementen alle Informationen, die für die korrekte Darstellung der Objekte benötigt werden: die IDs der ShapeNet-Objekte sowie die Werte für Position, Rotation und Skalierung. Für jedes Objekt wird ermittelt, welches ShapeNet-Thumbnail die Rotation am besten darstellen kann, und anschließend wird dieses vom ShapeNet-Service angefordert. Zur Ermittlung der richtigen Größe der Bilder in der Szene werden die Objektdimensionen aus den ShapeNet-Informationen benötigt, die ebenfalls angefordert werden. Bevor ein Thumbnail in die Szene eingefügt wird, wird in einem Vorverarbeitungsschritt das eigentliche Objekt aus dem Bild ausgeschnitten, damit es ohne den weißen Hintergrund dargestellt werden kann. Sobald Größe und Transformationen der Thumbnails bekannt sind, werden die Eckpunkte ermittelt, damit eine Kollisionserkennung vorgenommen werden kann. Anschließend werden die Objekte in der Szene angeordnet.

Der SceneUIController speichert für jedes Objekt sämtliche Daten, die im Attributpanel angezeigt werden sollen: Neben den Transformationsdaten, die bereits vom SceneController ermittelt wurden, sind dies die URLs der zwei Thumbnails, der Objektname und die zugehörigen WN Lemmata.

Bei der Verarbeitung von Benutzeraktionen ist eine Kommunikation zwischen den einzelnen Klassen nötig. Wird ein Token ausgewählt, soll das entsprechende Objekt auch in der Szene ausgewählt werden und umgekehrt. Bei der Auswahl eines Objekts/eines Tokens muss das entsprechende Attributpanel angezeigt und mit Werten gefüllt werden. Wird ein Objekt transformiert, müssen die Werte aktualisiert werden. Hierfür sind die im Diagramm sichtbaren, mit *on* beginnenden, Methoden der jeweiligen Klassen zuständig.

Beispiel Objektauswahl in der Szene: In diesem Fall ruft der SceneController die Methode *onSceneObjectSelected()* des SemAFPanelControllers auf. Dieser zeigt daraufhin in *toggleAttributePanel()* das entsprechende Attributpanel an und übergibt in einem Aufruf der Methode *setShapeNetPanelInfo()* des SceneUIControllers eine Referenz auf das ShapeNet-Tab, sodass dieser es mit Werten füllen kann.

Im Anchluß an jede Transformation werden die veränderten Daten des transformierten Objekts an das Back-End geschickt. Nachdem die darauffolgende Antwort mit dem aktualisierten CAS-Dokument empfangen wurde, werden alle Objekte entsprechend der erhaltenen Daten neu angeordnet. Dadurch ist die Konsistenz zwischen client- und serverseitigem Modell immer gesichert. Der erwartete Fall ist, dass die vom Back-End erhaltenen Positionsdaten mit den lokalen Daten übereinstimmen, sodass die Objektpositionen nicht verändert werden müssen und die Serverkommunikation für den Benutzer nicht bemerkbar ist.

4.3.3 UIMA Typsystem

Damit die im SemAF-Annotator erzeugten Szenen kompatibel zum VAnnotatoR sind, wurde das bereits für die Szenenerstellung im VAnnotatoR definierte UIMA Typsystem übernommen. Es werden nur vier Eigenschaften benötigt, um die Szenen zu speichern: Wird ein ShapeNet-Objekt mit einer ISOSpace-Entität verknüft, so wird die eindeutige ID, die für jedes Objekt definiert ist, in der Eigenschaft *object_id* des Elements hinterlegt. In den Eigenschaften *position*, *scale* und *rotation* werden Referenzen auf drei- bzw. vierdimensionale Vektor-Elemente (*Vec3* bzw. *Vec4*) gespeichert, in denen die eigentlichen Werte angegeben

```
<isospace:SpatialEntity xmi:id="10725" begin="10" end="12"
    object_id="9abd3d957518fd38cde56d3f260dd4f3" position="11270"
    rotation="11192" scale="11285"/>
<IsoSpatial:Vec3 xmi:id="11270" x="1.582" y="1.0582" z="-1.222"/>
<IsoSpatial:Vec4 xmi:id="11192" x="-0.659" y="0.254" z="0.254" w="0.659"/>
<IsoSpatial:Vec3 xmi:id="11285" x="1.276" y="1.276" z="1.276"/>
```

Abbildung 4.7: Objektzuordnung und Transformationsdaten im CAS-Dokument

sind. Abbildung 4.7 zeigt ein Beispiel. Die Rotationswerte liegen als Quaternion vor und müssen daher in ein im Zweidimensionalen interpretierbares Format umgerechnet werden.

4.3.4 Anordnung der Objekte

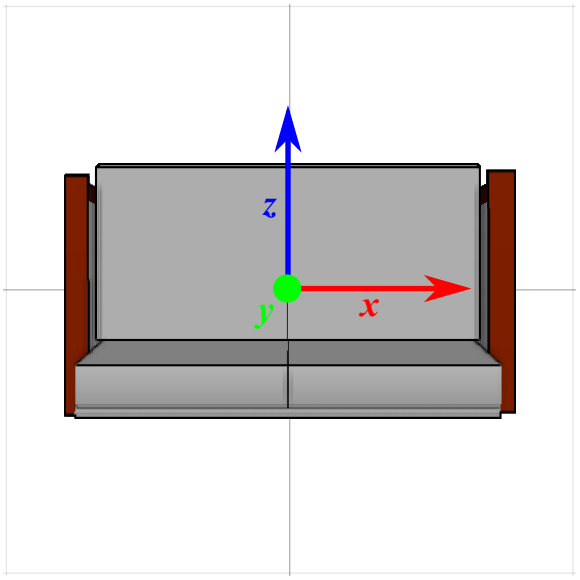
Der Aufbau des Koordinatensystems wurde ebenfalls mit dem VAnnotatoR abgeglichen, um eine konsistente Darstellung zu erreichen. Abbildung 4.9a zeigt, wie Objektkoordinatensysteme im VAnnotatoR definiert sind: Für alle Objekte zeigt die Vorderseite in Richtung der z-Achse und die Oberseite in Richtung der y-Achse. In der Ausgangsrotation $(0, 0, 0)$ stimmt das Objektkoordinatensystem mit dem globalen Koordinatensystem der Szene überein. Der Ursprung liegt immer im Mittelpunkt des Objekts.

Position: Das Koordinatensystem von SVG-Bildern ist in Abbildung 2.2 sichtbar. Der Ursprung liegt in der linken oberen Ecke und die y-Achse zeigt nach unten. Dasselbe gilt für mittels *<image>* eingefügte Bilder. Die in den CAS-Elementen gespeicherten Koordinaten müssen deshalb umgerechnet werden. Abbildung 4.8 zeigt, wie dies in einer Methode der Klasse Grid umgesetzt ist. Das Raster ist so konstruiert, dass der Ursprung an der Pixelposition $(0, 0)$ liegt, so dass hier keine zusätzliche Translation notwendig ist. Damit die z-Achse der Szene nach oben zeigen kann, wird der y-Koordinate des Thumbnails die negative z-Koordinate des Objekts zugewiesen und alle Thumbnails werden vertikal gespiegelt.

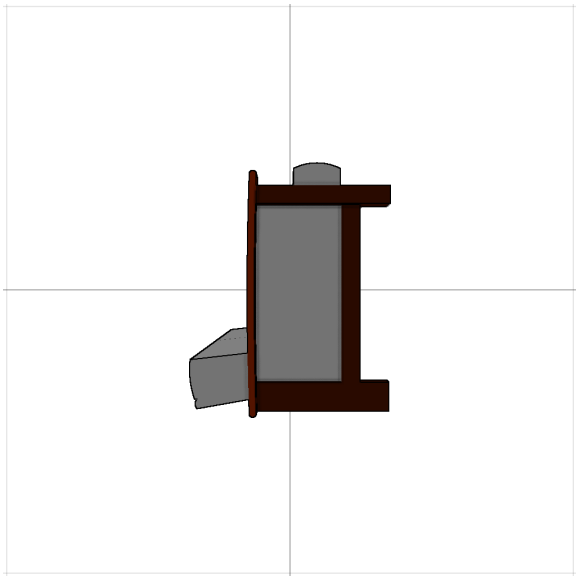
```
getGridCoordinates: function(x, z, xLength, zLength) {
    return {
        x:  x / this.stepUnit * this.stepPixels - 0.5*xLength,
        y: -z / this.stepUnit * this.stepPixels - 0.5*zLength
    }
},
```

Abbildung 4.8: Umrechnung der Koordinaten

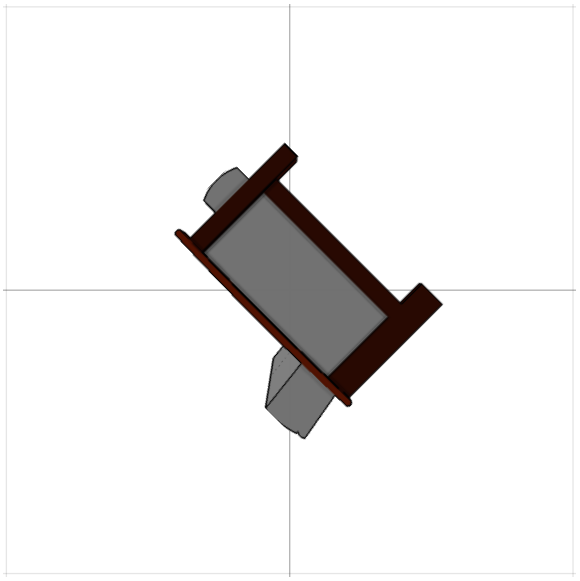
Rotation: Solange Objekte nur um die y-Achse rotiert werden, ist dies problemlos durch eine stufenlose Rotation des Thumbnails mit Hilfe von SVG-Funktionen möglich. Um aber auch Objekte darstellen zu können, die auf den Kopf oder auf die Seite gestellt wurden, müssen die



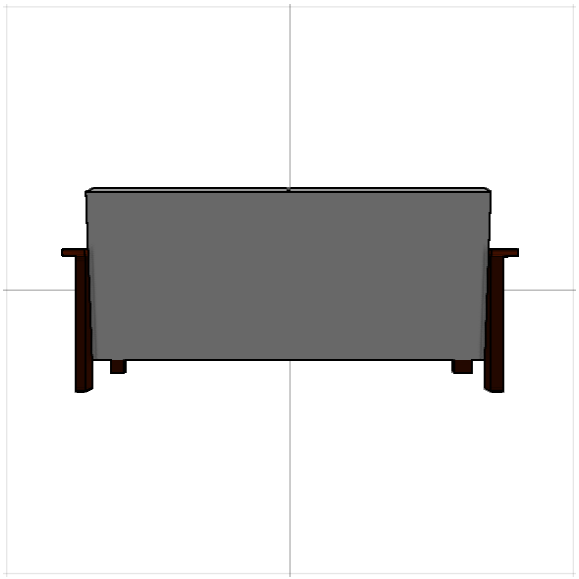
(a) Rotation $(0, 0, 0)$



(b) Rotation $(0, 0, 90)$



(c) Rotation $(0, -45, 90)$



(d) Rotation $(60, 0, 0)$

Abbildung 4.9: Verschiedene Objektrotationen

Rotationswerte interpretiert und ein anderes Thumbnail ausgewählt werden, das das Objekt aus der entsprechenden Perspektive zeigt. Hierzu wurde zunächst mit dem Unity3D-Editor ermittelt, welche Rotationen in den einzelnen Thumbnails repräsentiert sind. Es werden nur die ersten sechs Thumbnails aus Abbildung 2.1 verwendet. Die im CAS-Element gespeicherten Werte müssen vom Quaternion-Format in das Euler-Winkel-Format umgewandelt werden, damit sie sich im Zweidimensionalen interpretieren lassen. Eine Schwierigkeit hierbei ist, dass die Darstellung im Euler-Winkel-Format oft nicht eindeutig ist, sodass dieselbe Rotation mehrere Repräsentationen haben kann. Für die Umwandlung wird eine Funktion aus dem frei verwendbaren Javascript-Bibliothek THREE.js[1] benutzt. Anschließend werden die Rotationswerte um die x- und z-Achse betrachtet, um das Thumbnail auszuwählen, das der Rotation des Objekts am ehesten entspricht. Die y-Achse muss nicht beachtet werden, da hier die richtige Rotation wie oben beschrieben durch Rotieren des Thumbnails erreicht werden kann. Beispielsweise wurde die Couch in Abbildung 4.9b um $90°$ um die z-Achse gedreht. Das geeignete Thumbnail für diese Rotation ist das erste aus Abbildung 2.1, welches allerdings die Rotation $(0, -90, 90)$ darstellt. Deswegen wird dieses Thumbnail (nachdem es vertikal gespiegelt wurde) um $90°$ gedreht, wodurch die richtige Rotation erreicht wird.

Auch wenn die y-Achse des Objekts in Abbildung 4.9b nach links zeigt, resultiert eine anschließende Rotation um die y-Achse in Abbildung 4.9c. Das liegt daran, dass Rotationen immer in der Reihenfolge y-x-z ausgeführt werden. Die Rotation um die y-Achse entspricht somit immer dem Drehen des Thumbnails. Abbildung 4.9d zeigt einen Fall einer nichtachsenparallelen Rotation. Da diese durch kein Thumbnail korrekt dargestellt werden kann, wird stattdessen das nächstbeste angezeigt. Ab einem Wert von $(44, 0, 0)$ würde die Couch wieder von oben gezeichnet werden.

Größe: Zunächst hat jedes Thumbnail die Größe 512x512px. Um die Objekte innerhalb einer Szene mit korrekten relativen Größen darstellen zu können, sind im ShapeNet Datensatz Informationen über die Objektdimensionen (Breite, Tiefe und Höhe) hinterlegt. Diese müssen je nach Rotation des Objekts anders zum Skalieren des Thumbnails angewendet werden: In Abbildung 4.9a bestimmt z.B. die Objektbreite auch die Thumbnailbreite, und die Objekttiefe bestimmt die Thumbnailhöhe. In Abbildung 4.9b muss der Breite des Thumbnails dagegen die Objekthöhe, und der Höhe die Objekttiefe zugewiesen werden. Nach der Ermittlung dieser Initialgröße wird das Thumbnail mit dem Wert skaliert, der in der Eigenschaft *scale* des CAS-Elements gespeichert ist.

4.3.5 Kommunikation mit dem ShapeNet-Service

Die Kommunikation mit dem ShapeNet-Service ist in der Klasse *ShapeNetHelper* gekapselt. Sowohl der Scene- als auch der SceneUIController benutzen diese Klasse, um Daten vom ShapeNet-Webservice anzufordern. Intern benutzt der ShapeNetHelper die Klassen *XMLHttpRequest*[2] und *Promise*[3] der Javascript-API, um die Daten asynchron anzufragen. Folgende

[1] `https://threejs.org/`

[2] Ermöglicht das Nachladen von Daten via HTTP, ohne dass die gesamte Website neu geladen werden muss.

[3] Promise-Objekte werden stellvertretend für das Resultat einer asynchronen Operation verwendet. Die Methode *then()* eines Promise-Objekts erwartet eine Referenz auf eine Callbackfunktion, die aufgerufen wird, sobald das Ergebnis vorliegt.

Anfragen werden benutzt:

1. *search?search=[Suchstring]*

   Bei dieser Anfrage werden Informationen zu allen ShapeNet-Objekten zurückgegeben, deren Name den angegebenen Suchstring enthält. Unter diesen Informationen ist auch eine eindeutige ID, die für die nachfolgenden Anfragen benötigt wird. Die Rückgabe der Daten erfolgt im JSON-Format. Der SceneUIController benutzt diese Anfrage, um dem Benutzer bei der Objektauswahl die für den Suchbegriff relevanten Objekte anzuzeigen.

2. *getFeature?id=[ID des ShapeNet-Objekts]*

   Diese Anfrage gibt Informationen zu einem Objekt zurück, dessen ID bereits bekannt ist. Sie wird sowohl vom Scene- als auch vom SceneUIController benutzt, um die Daten von Objekten zu erhalten, die bereits ausgewählt wurden. Der SceneController benötigt aus diesen Daten die Größeninformationen, die die Breite, Tiefe und Höhe der Objekte beinhalten, um die Objekte mit der richtigen Größe anzeigen zu können. Der SceneUIController verwendet diese Daten, um den Namen und die WordNet Lemmata eines Objekts im Attributpanel anzuzeigen.

3. *getThumbnail?id=[ID des ShapeNet-Objekts]&pic=[ID des Thumbnails].*

   Diese Anfrage liefert das Thumbnail mit der angegebenen Thumbnail-ID des Objekts mit der angegebenen Objekt-ID. Der SceneController benutzt diese Anfrage, um die Thumbnails im Attributpanel anzuzeigen, der SceneUIController, um die Thumbnails in der Szene anzuzeigen.

Die obigen Anfragen sind in den Methoden *getSearchData()*, *getData()* und *getThumbnail()* gekapselt. Der ShapeNetHelper speichert alle angeforderten Daten. Beim Aufruf einer der o.g. Methoden wird geprüft, ob die Daten bereits lokal vorliegen. Falls das der Fall ist, oder falls bereits eine Anfrage gesendet wurde, die noch nicht beantwortet wurde, wird das entsprechende erfüllte bzw. noch offene Promise-Objekt zurückgegeben. Andernfalls wird eine neue Anfrage an den ShapeNet-Service gesendet und das offene Promise-Objekt zurückgegeben.

4.3.6 Vorverarbeitung der Thumbnails

Im Vorverarbeitungsschritt wird der weiße Hintergrund der Thumbnails entfernt (Abbildung 4.10b). Dieser Schritt erfüllt nicht nur ästhetische Zwecke (die weißen Hintergründe würden die Rasterlinien und andere Objekte in unschöner Art und Weise überlagern) sondern ist essentiell, um die Objekte in ihrer richtigen Größe darstellen zu können.

Die Objekte haben in ihren Thumbnails eine individuelle Skalierung, die für die Darstellung in einer Szene nicht geeignet ist, da die Proportionen der Objekte zueinander nicht stimmen würden. Die für die Darstellung in der Szene richtigen Größeninformationen werden in den ShapeNet-Daten des Objekts zur Verfügung gestellt und können über den ShapeNet-Service abgefragt werden. Abbildung 4.10a zeigt Objekte in ihrer ursprünglichen Größe im Thumbnail und in der richtigen Größe für die Eingliederung in die Szene.

Die Größeninformationen können jedoch nur richtig auf ein Objekt angewendet werden, wenn es vorher aus dem Thumbnail ausgeschnitten wurde. Andernfalls könnte nur das ganze Thumbnail skaliert werden, was erstens zur Folge hätte, dass die Größe des Objekts in der Szene nicht durch das Objekt im Thumbnail, sondern durch das ganze Thumbnail inklusive weißem Rand, repräsentiert werden würde, und zweitens, dass das Objekt unter Umständen verzerrt werden würde (Abbildung 4.10c).

Vorgehensweise

Nachdem ein Thumbnail vom ShapeNet-Service erhalten wurde, wird es in ein unsichtbares HTML-*<canvas>*-Element geladen, in dem die pixelbasierte Bearbeitung von Bildern möglich ist. Daraufhin wird die Bounding-Box des Objekts im Bild bestimmt, indem, beginnend an den Rändern des Bildes, über alle Pixel in Richtung Mitte iteriert wird. Sobald ein Pixel gefunden wurde, dessen Farbwerte nicht der weißen Hintergrundfarbe entsprechen, wurde eine Seite der Bounding-Box bestimmt. Zuerst wird spaltenweise von links nach rechts sowie rechts nach links iteriert, um die linke bzw. rechte Seite der Bounding-Box zu bestimmen, nachfolgend zeilenweise von oben nach unten bzw. unten nach oben, um die obere bzw. untere Seite zu erhalten. Anschließend wird das Bild auf die Bounding Box zugeschnitten.

Im nächsten Schritt werden die verbleibenden weißen Hintergrundpixel transparent gemacht. Da es häufiger vorkommt, dass die Objekte selbst weiße Teile enthalten, kann diese Operation nicht einfach für alle weißen Pixel durchgeführt werden. Stattdessen kommt an dieser Stelle eine abgewandelte Version des in (Vincent und Soille 1991) vorgestellten Algorithmus zum Finden von Zusammenhangskomponenten in Graphen zum Einsatz[1]. Der $O(n)$-Algorithmus findet alle weißen Pixel, die mit dem Rand des Bildes verbunden sind, und setzt diese transparent. Der Pseudocode ist in Abbildung 4.11 dargestellt. Da es unmöglich ist, weiße Hintergrundpixel, die innerhalb der Objektgrenzen liegen, von weißen Pixeln, die zum Objekt gehören, zu unterscheiden, bleiben diese weiß.

Wie lange die Vorverarbeitung dauert, hängt stark von der Beschaffenheit des Thumbnails ab. In der Praxis wurde ein durchschnittlicher Wert von etwa 30ms pro Bild gemessen[2].

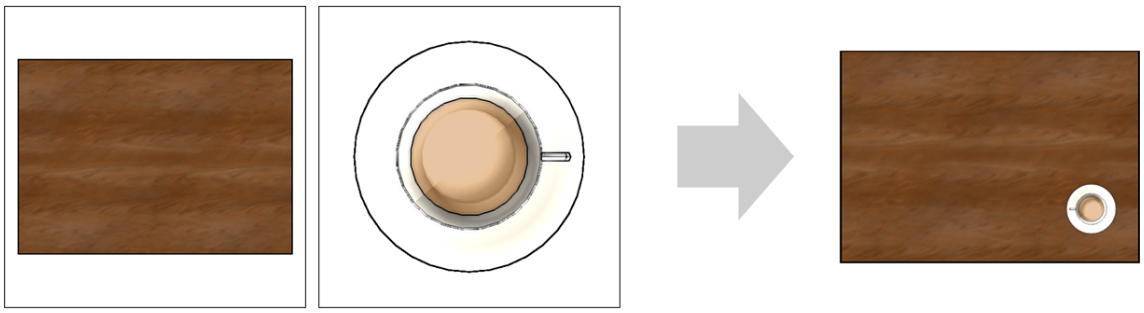
4.3.7 Kollisionserkennung

Eine Kollisionserkennung wird benötigt, um Objekte, die andere verdecken, transparent zeichnen zu können und um Objekte aufeinander platzieren zu können.

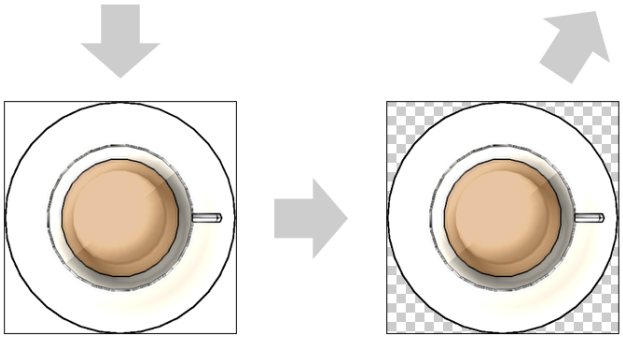
Erste Ansätze: Zunächst wurde versucht, die Kollisionserkennung mit nativen Funktionen der SVG- bzw. DOM-API zu implementieren. Die SVG-Funktion *checkIntersection(SVGRect r, SVGElement el)* erwartet zwar ein achsenparalleles Rechteck als erstes Argument, hätte mit einem kleinen Trick aber trotzdem verwendet werden können, um auch Überschneidungen zwischen rotierten Objekten zu erkennen. Wie sich herausstellte, ist diese Funktion allerdings nicht sehr performant und wird von Firefox nicht unterstützt. Die DOM-Funktion *elementsFromPoint(x, y)* arbeitet dagegen deutlich schneller und ist in allen gängigen Browsern implementiert. Sie liefert eine Liste aller DOM-Elemente, die den angegebenen Punkt

[1] S. a. `https://en.wikipedia.org/wiki/Connected-component_labeling`

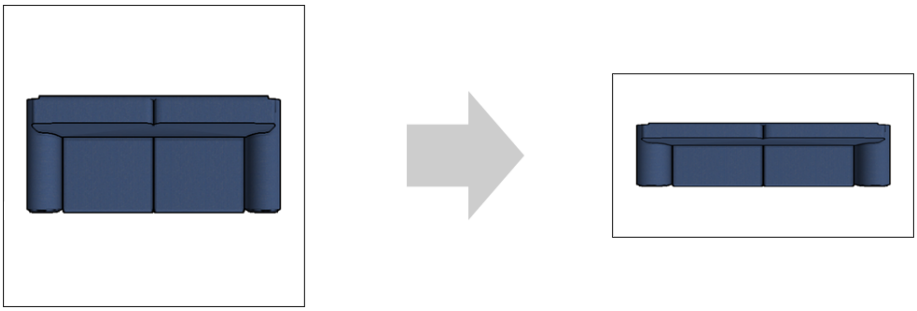
[2] Testsystem: AMD Ryzen 3600, 16 GB RAM, Windows 10, Google Chrome.



(a) Die Thumbnails in ihrer initialen Form und das Endergebnis in der Szene



(b) Die zwei Vorverarbeitungsschritte: Zuschneiden des Thumbnails und Entfernen des verbleibenden Hintergrunds



(c) Skalieren eines Thumbnails ohne vorheriges Zuschneiden

Abbildung 4.10: Vorverarbeitung der Thumbnails (schwarzer Rand zur Verdeutlichung hinzugefügt).

```
1  var bgPixels = []
2  var visited = []
3
4  for (jeden Pixel p am Bildrand) {
5      if (!visited[p]) {
6          visited[p] = true
7          if (p ist weiß) {
8              bgPixels.push(p)
9              Setze den Alpha-Wert von p auf 0.
10             Nachbarsuche()
11         }
12     }
13 }
14
15 function Nachbarsuche() {
16     while (bgPixels.length > 0) {
17         p = bgPixels.pop()
18         for (jeden benachbarten (oberhalb, unterhalb, links und rechts von
               p) Pixel q) {
19             if (!visited[q]) {
20                 visited[q] = true
21                 if (q ist weiß) {
22                     bgPixels.push(q)
23                     Setze den Alpha-Wert von q auf 0.
24                 }
25             }
26         }
27     }
28 }
```

Abbildung 4.11: Algorithmus zum Entfernen des Thumbnailhintergrunds

enthalten, sodass eine Überschneidung von Objekten durch Testen der Eckpunkte gefunden werden könnte. Hier ergab sich aber, dass die Funktion nur Elemente zurückgibt, die sich innerhalb des Viewports[1] befinden. Eine Implementierung mit dieser Funktion wurde daher als nicht robust genug empfunden.

Lösung: Deswegen wurde stattdessen eine rein mathematische Lösung gewählt. Da a priori nur die Mittelpunkte der 3D-Objekte und die Koordinaten der zur Positionierung in der Szene ermittelten linken oberen Ecken der untransformierten Thumbnails (vgl. Unterabschnitt 4.3.4) bekannt sind, müssen zunächst die Eckpunkte der skalierten und rotierten Thumbnails berechnet werden, bevor diese im nächsten Schritt benutzt werden können, um mittels des Separating Axis Theorems[2] die Überschneidungen zu berechnen. Die Performanz dieser Lösung ist noch etwas besser als die des versuchten Ansatzes mit der *elementsFromPoint*-Funktion. Beispielsweise benötigt die Kollisionserkennung für die Szene in Abbildung 4.1 gerade einmal 0.2ms[3]. Sollte sich in der Praxis herausstellen, dass die Erkennung von minimalen Kollisionen mit nur einem Eckpunkt nicht zweckdienlich ist, könnte die Funktion ohne allzu großen Aufwand so abgeändert werden, dass nur Überschneidungen mit zwei oder mehr Eckpunkten oder dem Mittelpunkt eines Objekts erkannt werden.

4.4 Benutzerinteraktionen

4.4.1 Hinzufügen und Entfernen von Objekten

Möchte der Benutzer einem Token ein ShapeNet-Objekt zuweisen, so muss er diesen zuerst anwählen, anschließend im Attributpanel das ShapeNet-Tab auswählen und dort auf „add object“ klicken. Es erscheint ein Fenster, in dem Thumbnails der ShapeNet-Objekte auswählbar sind (Abbildung 4.4). Sobald der Benutzer auf eines davon anklickt, wird das Fenster geschlossen, das Objekt wird mit der Startposition $(0, Objekthöhe/2^4, 0)$ zur Szene hinzugefügt und automatisch ausgewählt. Das Entfernen eines Objekts ist entweder durch Klicken auf den Button „remove Object“ oder durch Drücken der Entf-Taste möglich.

4.4.2 Objekttransformationen

Bewegt der Benutzer den Mauszeiger über ein Objekt, verändert sich der Mauszeiger und das Objekt wird etwas aufgehellt. Wird ein Objekt durch einen Mausklick ausgewählt, so wird eine Bounding Box um das Objekt herum gezeichnet (Abbildung 4.12a). Um ein versehentliches Verschieben zu verhindern, muss ein Objekt erst einmal ausgewählt werden, bevor es sich durch Ziehen mit der Maus verschieben lässt.

An den Ecken der Bounding Box sind aus Bildbearbeitungsprogrammen oder 3D-Editoren bekannte Griffe in Form von Quadraten angebracht, die mit der Maus gezogen werden können, um das Objekt zu vergrößern oder zu verkleinern [5]. Beim Berühren der Quadrate ver-

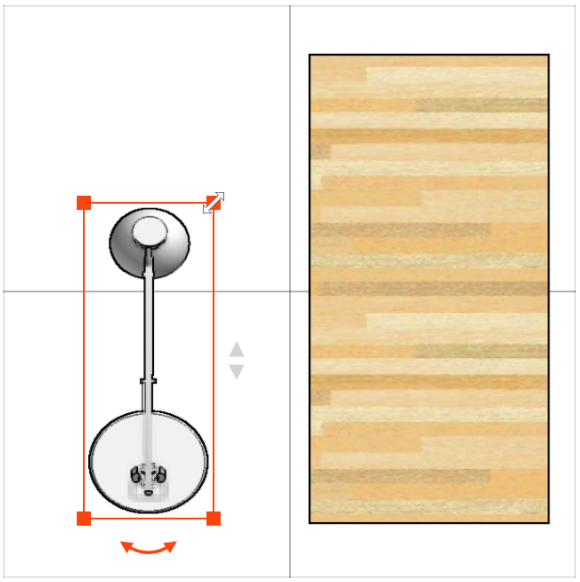
[1] Der in der aktuellen Scrollposition sichtbare Bereich einer Webseite.

[2] S. z.B. `https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection`

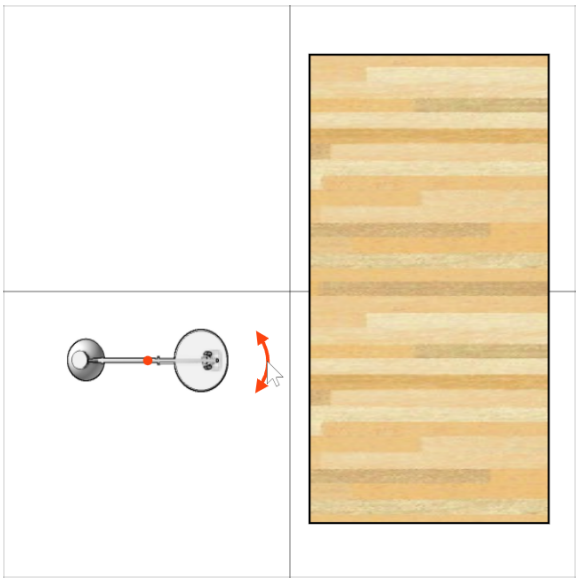
[3] Testsystem: AMD Ryzen 3600, 16 GB RAM, Windows 10, Google Chrome.

[4] Entspricht der Platzierung auf dem Boden. Vgl. Unterabschnitt 4.3.4

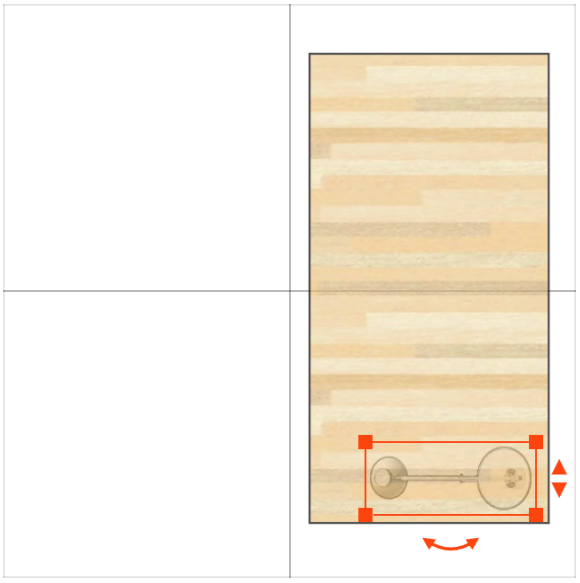
[5] Über jedem Bedienelement ist ein etwas größeres, unsichtbares Rechteck gezeichnet, das mit der Maus interagiert, damit die Elemente leichter klickbar sind.



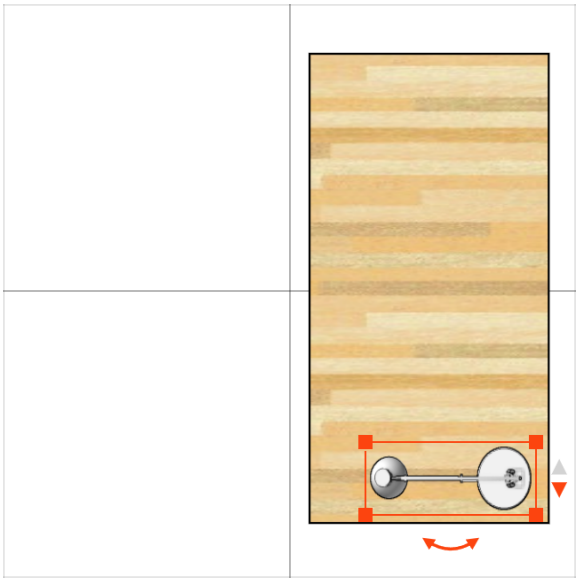
(a) vor dem Skalieren



(b) nach der Skalierung, während der Rotation



(c) nach dem Verschieben



(d) nach der vertikalen Positionierung

Abbildung 4.12: Positionieren eines Objekts mit Hilfe von Transformationen

wandelt der Mauszeiger sich in einen schrägen Doppelpfeil, der anzeigt, in welche Richtung die Maus bewegt werden muss, um eine Skalierung zu erreichen. Da nur die proportionale Skalierung möglich sein soll, ist das Skalieren nur an den Ecken, nicht an den Seiten der Bounding Box, möglich.

Unterhalb der Bounding Box ist ein gerundeter Doppelpfeil angezeigt, mit dem ein Objekt durch Ziehen mit der Maus rotiert werden kann. Der Ursprung der Rotation liegt wie auch der Ursprung der Skalierung immer im Mittelpunkt des Objekts. Während jeder Transformation werden die Bedienelemente für die jeweils anderen Transformationen kurz ausgeblendet: Während der Rotation ist nur der Rotationspfeil und der Mittelpunkt der Rotation sichtbar (Abbildung 4.12b). Nach Abschluß der Rotation wird eine neue Bounding Box achsenparallel gezeichnet.

Durch die Pfeile rechts von der Bounding Box lassen sich Objekte vertikal positionieren. Die ausgegrauten Pfeile in Abbildung 4.12a deuten an, dass die Lampe sich auf dem Boden befindet und sich mit keinem höher liegenden Objekt schneidet. In Abbildung 4.12c wurde die Lampe auf die Position des Schreibtisches verschoben. Der rote nach-oben-Pfeil zeigt an, dass sie sich aber im Moment noch unter dem Schreibtisch befindet[1], der deshalb transparent gezeichnet wird. Der rote *nach-unten*-Pfeil zeigt an, dass die Lampe nach der Skalierung, da auch in Richtung der y-Achse skaliert wurde, nicht mehr auf dem Boden steht. Nach Klicken des nach-oben-Pfeils wird die Lampe auf dem Schreibtisch positioniert, der Pfeil wird ausgegraut und die Transparenz des Schreibtisches verschwindet (Abbildung 4.12d). Durch Klicken des nach-unten-Pfeils könnte die Lampe wieder auf den Boden gestellt werden.

4.4.3 Transformationen über das Attributpanel

Das Attributpanel bietet ebenfalls einige Möglichkeiten, um Objekte zu transformieren: Durch Verändern der Werte für Skalierung, Positionierung in x- oder z-Richtung, oder Rotation um die y-Achse können Objekte in gleicher Weise transformiert werden, wie das auch direkt in der Szene mittels der Bounding Box des ausgewählten Objekts möglich ist. Die Werte können entweder per Tastatur eingegeben oder mit den Pfeilsymbolen oder dem Mausrad schrittweise verändert werden. Um nicht unnötig viele Update-Nachrichten (nach jedem Klick eines Pfeilsysmbols, oder nach jedem „Klick“ des Mausrads) an das Back-End zu senden, wird die Transformation zunächst nur lokal ausgeführt und erst an das Back-End geschickt, wenn das Formularelement den Fokus verloren hat. Wird ein Objekt direkt in der Szene transformiert, so werden umgekehrt die Werte in den Formularelementen „live“ aktualisiert.

Zusätzlich erlauben Formularelemente die Rotation um die x- und z-Achse der Objekte, die in der Szene selbst nicht möglich ist. In diesen Formularelementen ändern die Pfeilsymbole die Werte immer in 90°-Schritten. Mit der Tastatur können theoretisch auch andere Winkel angegeben werden, die aber natürlich, aufgrund der zweidimensionalen Darstellung, nicht visualisiert werden können. Da die Rotationen in der für Menschen besser lesbaren Euler-Winkel Repräsentation angezeigt werden, kann es bei der Rotation um die x- und z-Achse zu dem bei dieser Repräsentation bekannten Problem des Gimbal-Locks kommen: Nach Drehen

[1] Genauer gesagt ist der nach-oben-Pfeil aktiviert, weil die unterste Fläche der Lampe unter der obersten Fläche des Schreibtisches liegt.

um die x-Achse kann es passieren, dass die y- und z-Achsen in dieselbe Richtung zeigen, sodass Rotationen um beide dieser Achsen verwirrenderweise den gleichen Effekt haben.

Deshalb ist es alternativ möglich, die Rotation eines Objekts durch Klicken auf das rechte Thumbnail im Attributpanel zu verändern. Es erscheint ein kleines Fenster, das die sechs möglichen Rotationen anzeigt (Abbildung 4.3 rechts). Durch Klicken eines dieser Thumbnails wird die entsprechende Änderung an das Back-End geschickt und die Rotation wird in der Szene übernommen.

4.4.4 Auswahl von verdeckten Objekten

Bedingt durch die zweidimensionale Darstellung aus der Vogelperspektive kann es vorkommen, dass ein Objekt von einem anderen, wie bereits gesehen, verdeckt wird und daher erstens nicht sichtbar und zweitens nicht auswählbar ist. Deswegen wurde eine Funktion implementiert, die es dem Benutzer erlaubt, durch Drücken der „t“-Taste das weiter oben liegende Objekt transparent erscheinen zu lassen. Gleichzeitig wird es ermöglicht, durch das transparente Objekt durchzuklicken und so das untere Objekt auszuwählen. Alternativ, ist es möglich das verdeckte Objekt durch Klick auf den entsprechenden Token auszuwählen, was ebenfalls dazu führt, dass das höhere Objekt transparent gezeichnet wird.

In einigen Fällen kann es passieren, dass ein Objekt schwer sichtbar ist, weil es ähnliche Farben wie das darüber oder darunter liegende Objekt hat. Um diese Objekte in der Szene besser identifizieren zu können, werden durch Drücken der t-Taste auch die Bounding Boxen aller Objekte angezeigt. Nach erneutem Drücken der t-Taste verschwinden beide Effekte wieder, es sei denn, es wurde ein verdecktes Objekt ausgewählt. In diesem Fall bleibt das höhere Objekt transparent, bis das verdeckte Objekt wieder abgewählt wurde.

5 Nachbesserungen und neue Annotationstypen

5.1 Neue Annotationstypen

Zwei neue Relationstypen wurden eingefügt: der MetaLink (vgl. Abschnitt 2.2) und der SR-Link aus dem SemAF-SR-Standard (vgl. Abschnitt 2.1). Für MetaLinks kann einer der drei Untertypen COREFERENCE, SUBCOREFERENCE, und SPLITCOREFERENCE ausgewählt werden. Wird mittels eines COREFERENCE-MetaLinks auf einen Token verwiesen, dem ein ShapeNet-Objekt zugeordnet ist, dann wird dieses auch dem anderen Token zugewiesen, sodass es in der Szene über beide Tokens anwählbar ist.

5.2 Umorganisation des Optionspanels

Da durch die neuen Links das Panel unter „Create Element“ etwas zu voll geworden wäre, wurde diese Gelegenheit genutzt, um die Funktion zum Erstellen von leeren Entitäten zu überarbeiten: Die Typ-Buttons zum Erstellen von leeren Entitäten (vgl. Abschnitt 3.2.4) wurden entfernt. Stattdessen können leere Entitäten nun mit dem Button „Create ∅-Enitty“ erzeugt werden und anschließend können die Buttons unter „Entity Types“ wie für normale Entitäten benutzt werden, um einen Typ zuzuweisen. Diese neue Aufteilung ist für den Benutzer vermutlich ohnehin intuitiver. Wird nun ein Token ausgewählt, dann sind jetzt immer die Buttons unter „Entity Types“ für das Anzeigen und die Änderung des Typs zuständig. In der vorherigen Umsetzung variierte die Zuständigkeit je nachdem, ob es sich um eine Text- oder leere Entität handelte.

5.3 Lesbarkeit von Relationspfeilen

Im SemAF-Annotator 1.0 gab es zwar schon eine Funktion, die die Überlagerung von Link-Pfeilen verhindert (s. Abbildung 5.1), indem einer der Pfeile nach oben versetzt gezeichnet wird. Das funktionierte aber nur für einzeilige Links und war eigentlich für Fälle wie in Abbildung 5.1 sichtbar gedacht, in denen nicht mehrere Links dieselben zwei Tokens verbinden. Bei ISOSpace kommt es jedoch häufig vor, dass sowohl eine topologische als auch eine Richtungsbeziehung zwischen zwei Objekten annotiert werden soll und somit zwei Tokens gleichzeitig mit einem QSLink und einem OLink verbunden werden müssen. Waren die zwei Tokens nicht in derselben Zeile, so führte dies wie in Abbildung 5.2 (links) sichtbar dazu, dass die Link-Pfeile sich komplett überlagerten, sodass auch nur einer der Pfeile anklickbar war. In dieser Arbeit wurde deswegen zusätzlich eine Funktion eingefügt, die solche Fälle erkennt und die Pfeile dann versetzt zeichnet, um eine bessere Lesbarkeit zu gewährleisten (Abbildung 5.2 rechts).

Abbildung 5.1: Vermeidung von Link-Überschneidungen bei einzeiligen Links

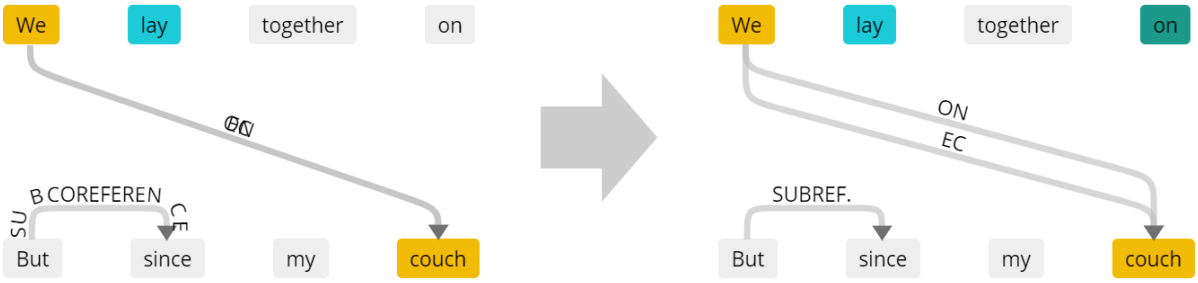


Abbildung 5.2: Vermeidung von Überschneidungen bei zeilenübergreifenden Links und Abkürzung von Relationstypen

Weiterhin wurde eine Funktionalität zur Abkürzung der angezeigten Relationstypen eingefügt (Abbildung 5.2). In Version 1.0 des SemAF-Annotators konnte es dazu kommen, dass der Text eines Relationstyps zu lang für die ordentliche Darstellung über einem Link-Pfeil war. Die neue Funktionalität kürzt den Text in solchen Fällen auf eine angemessene Länge. Für vordefinierte Relationstypen können in den Typdefinitionen im SemAFPanelModel kurze Varianten angegeben werden (Eigenschaft *shortValues*), die dann, falls notwendig, den vollständigen Relationstyp ersetzten. Ist dagegen der Text eines frei wählbaren Relationstyps zu lang, so wird dieser iterativ um einen Buchstaben gekürzt, bis eine ausreichend kurze Länge erreicht wurde. In den seltenen Fällen, dass ein vordefinierter, bereits gekürzter, Relationstyp zu lang ist, wird auf diesen dasselbe Verfahren angewendet.

6 Fazit

In dieser Arbeit wurde eine Funktionalität zur Erstellung von Szenen in der zweidimensionalen Weboberfläche des SemAF-Annotators entwickelt. Die Zweidimensionalität hat einerseits den Vorteil der einfachen Bedienbarkeit und der übersichtlichen, gleichzeitigen Darstellung von Textannotationen und Szenenobjekten. Anderseits gibt es auch einige Einschränkungen: Mit Ausnahme der freien Objektrotation um die y-Achse können nur achsenparallele Rotationen visualisiert werden. Eine präzise vertikale Ausrichtung der Objekte ist nach dem jetzigen Stand ebenfalls nicht ohne Weiteres möglich. Diese Einschränkungen waren aber bereits bei der Aufgabenstellung bekannt. Die entwickelte Funktionalität war deshalb als ergänzender Vorverarbeitungsschritt zu einer präziseren Bearbeitung im VAnnotatoR konzipiert.

Dennoch lassen sich in vielen Fällen bereits gute Ergebnisse erzielen. Abbildung 6.1 zeigt, wie zwei Szenen, die mit dem SemAF-Annotator erstellt wurden (links), im VAnnotatoR aussehen (rechts). Das Auswählen und anschließende Ausrichten von Objekten in der xz-Ebene ist wie erwartet problemlos möglich. Außerdem lässt sich ein Objekt, trotz der für diese Operation ungünstigen Perspektive, auf einem anderen platzieren. In der ersten Szene wurde ein Fernseher auf einer TV-Bank und eine Zeitschrift auf einem Tisch positioniert. In Szene 2 war es kein Problem, den Schreibtisch mit typischen Utensilien auszustatten. Das funktioniert allerdings nur, solange das Objekt auf der obersten Fläche des unteren Gegenstands platziert werden soll. Da dem System nur die Gesamthöhe der Objekte bekannt ist, können Katze und Person nicht genau auf der Sitzfläche der Couch platziert werden. Stattdessen schweben sie auf der Höhe der Rückenlehne. Genauso wäre es zwar möglich einen Gegenstand auf einem Regal, aber nicht auf einem Brett des Regals, abzulegen.

Weiterhin könnte es in einigen Fällen ein Problem sein, dass für den Benutzer nicht klar sichtbar ist, ob die Höhe eines ausgewählten Objekts für die Eingliederung in die Szene plausibel ist. Bei dem Tisch in der ersten Szene handelt es sich um einen „coffee table“, der deswegen niedrig genug ist, dass eine Platzierung zwischen Couch und TV-Bank Sinn macht. Es wäre aber denkbar, dass ein Benutzer an dieser Stelle einen Küchentisch auswählt, ohne sich über die Höhe Gedanken zu machen. Nach dem jetzigen Stand könnten die Höhen von Couch und Tisch nur verglichen werden, indem beide Gegenstände auf die Seite gedreht werden.

6.1 Zukünftige Arbeiten

Um die angesprochenen Probleme in den Griff zu bekommen, würde sich die Entwicklung einer zusätzlichen Sicht anbieten, die die Szene von der Seite zeigt, sodass die vertikalen Positionen und Höhen der Objekte erkennbar und veränderbar wären. Durch die zweidimensionale Darstellung wären aber natürlich auch diese Sichten nicht frei von Problemen. In Abbildung 6.2 wurde skizziert, wie die erste Szene aussehen würde, wenn die Kamera in

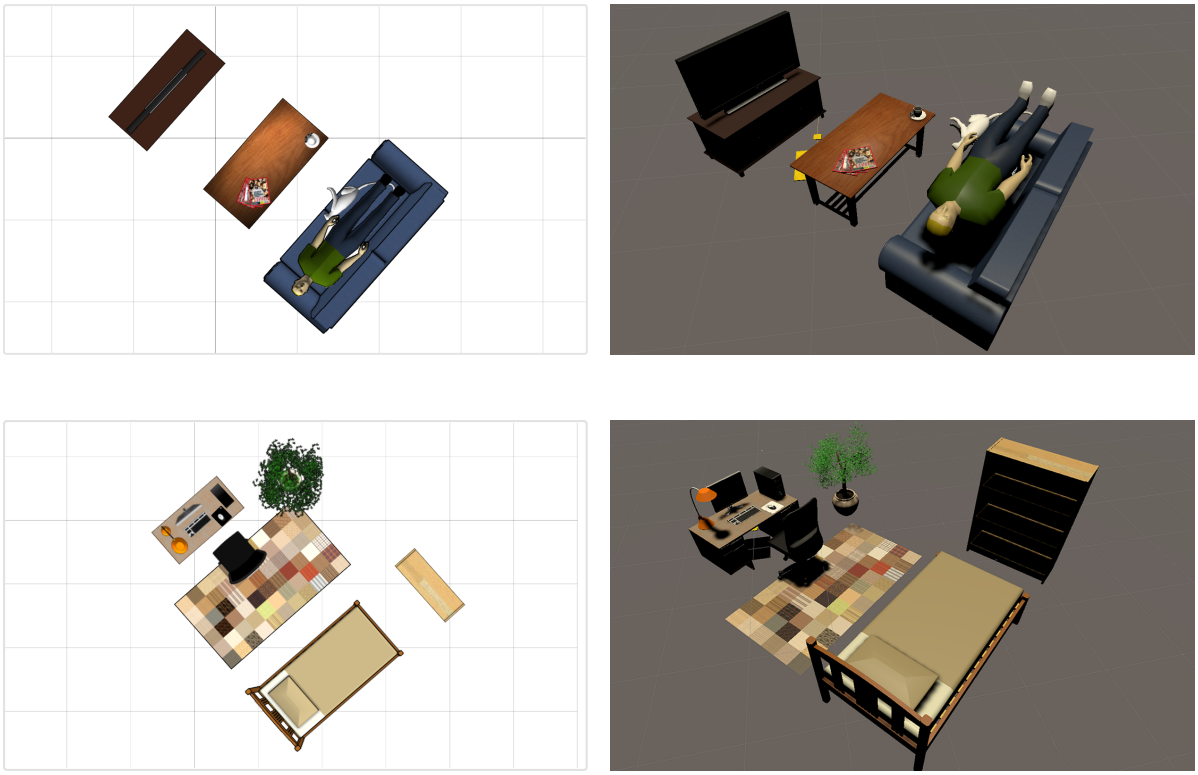


Abbildung 6.1: Zwei im SemAF-Annotator erzeugte Szenen im VAnnotatoR

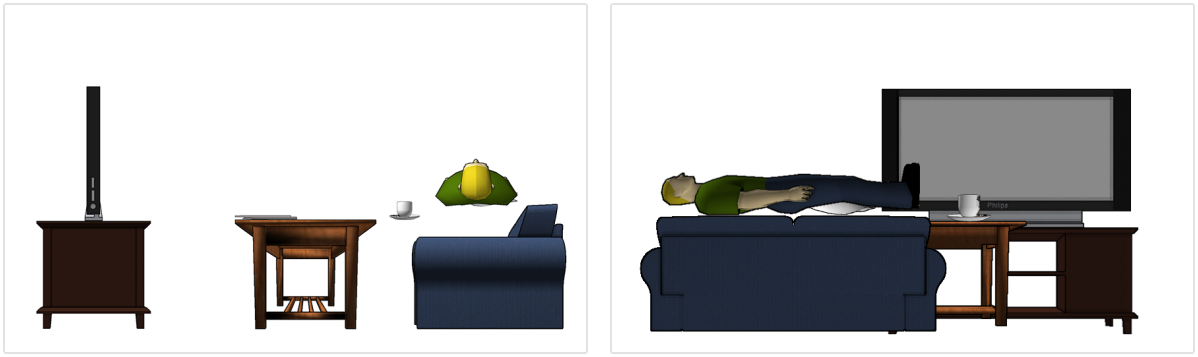


Abbildung 6.2: Skizzierte Seitenansicht einer Szene

Richtung der z-Achse (links) oder der negativen x-Achse (rechts) zeigen würde. Obwohl die Objekte schief in der Szene stehen, könnten sie wieder nur achsenparallel gezeichnet werden. Die sichtbaren Anordnungen werden erreicht, indem die Objekte individuell auf eine achsenparallele Ausrichtung gedreht werden. Das hat zur Folge, dass die relativen Positionen der Objekte zueinander nicht stimmen können, sodass die Tasse links nicht mehr auf dem Tisch steht und die Gegenstände im rechten Bild horizontal versetzt sind. Für den Benutzer könnte das etwas verwirrend sein.

Wie oft in der Praxis schräg stehende Gegenstände vorkommen werden, ist aber unklar. Wenn, wie bei diesem Beispiel, die gesamte Szene rotiert ist, könnte auch zuerst eine Achse ermittelt werden, die zu möglichst vielen Objekten in der Szene parallel ist, und die Darstellung dann entlang dieser Achse erfolgen. In diesem Fall würden die oben beschriebenen Probleme verschwinden.

Literatur

Abrami, Giuseppe, Alexander Henlein, Attila Kett und Alexander Mehler (2020). „Text2SceneVR: Generating Hypertexts with VAnnotatoR as a Pre-Processing Step for Text2Scene Systems“. In: *Proceedings of the 31st ACM Conference on Hypertext and Social Media*. HT '20. Virtual Event, USA: Association for Computing Machinery, S. 177–186. ISBN: 9781450370981. DOI: 10.1145/3372923.3404791. URL: https://doi.org/10.1145/3372923.3404791.

Abrami, Giuseppe und Alexander Mehler (2018). „A UIMA Database Interface for Managing NLP-related Text Annotations“. In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*. LREC 2018. Miyazaki, Japan.

Abrami, Giuseppe, Alexander Mehler, Andy Lücking, Elias Rieb und Philipp Helfrich (Mai 2019). „TextAnnotator: A flexible framework for semantic annotations“. In: *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation, (ISA-15)*. ISA-15. Gothenburg, Sweden.

Chang, Angel, Thomas Funkhouser u. a. (2015). „ShapeNet: An Information-Rich 3D Model Repository“. arXiv: 1512.03012 [cs.GR].

Chang, Angel, Manolis Savva und Christopher Manning (Juni 2014). „Interactive Learning of Spatial Knowledge for Text to 3D Scene Generation“. In: *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*. Baltimore, Maryland, USA: Association for Computational Linguistics, S. 14–21. DOI: 10.3115/v1/W14-3102. URL: https://aclanthology.org/W14-3102.

Coyne, Robert und Richard Sproat (Jan. 2001). „WordsEye: An automatic text-to-scene conversion system“. In: *Proceedings of SIGGRAPH 2001*, S. 487–496.

Eckart de Castilho, Richard u. a. (Dez. 2016). „A Web-based Tool for the Integrated Annotation of Semantic and Syntactic Structures“. In: *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*. Osaka, Japan: The COLING 2016 Organizing Committee, S. 76–84. URL: https://www.aclweb.org/anthology/W16-4011.

Ferrucci, David, Adam Lally, Karin Verspoor und Eric Nyberg (März 2009). „Unstructured Information Management Architecture (UIMA) Version 1.0“. OASIS Standard. URL: https://docs.oasis-open.org/uima/v1.0/uima-v1.0.html.

Gleim, Rüdiger, Alexander Mehler und Alexandra Ernst (2012). „SOA implementation of the eHumanities Desktop“. In: *Proceedings of the Workshop on Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts, Digital Humanities 2012, Hamburg, Germany*.

Hemati, Wahed, Tolga Uslu und Alexander Mehler (2016). „TextImager: a Distributed UIMA-based System for NLP“. In: *Proceedings of the COLING 2016 System Demonstrations*. Federated Conference on Computer Science und Information Systems. Osaka, Japan.

Ide, Nancy und James Pustejovsky (2017). „Handbook of Linguistic Annotation“. Springer. ISBN: 978-94-024-0881-2.

ISO (2014). „Language resource management — Semantic annotation framework (SemAF) — Part 4: Semantic roles (SemAF-SR)“. Standard ISO/IEC TR 24617-4:2014. URL: https://www.iso.org/obp/ui/#iso:std:iso:24617:-4:ed-1:v1:en.

– (2016). „Language resource management — Semantic annotation framework — Part 6: Principles of semantic annotation (SemAF Principles)“. Standard ISO/IEC TR 24617-6:2016. URL: https://www.iso.org/standard/60581.html.

– (2020). „Language resource management — Semantic annotation framework — Part 7: Spatial information (ISO-Space)“. Standard ISO/IEC TR 24617-7:2020. URL: https://www.iso.org/obp/ui/#iso:std:iso:24617:-7:ed-2:v1:en.

Ma, Rui u. a. (2018). „Language-driven synthesis of 3D scenes from scene databases“. In: *SIGGRAPH Asia 2018 Technical Papers*. ACM, S. 212.

Pustejovsky, James, Parisa Kordjamshidi u. a. (Jan. 2015). „SemEval-2015 Task 8: SpaceEval“. In: *Proceedings of the 9th International Workshop on Semantic Evaluation*, S. 884–894. DOI: 10.18653/v1/S15-2149.

Pustejovsky, James, Jessica L. Moszkowicz und Marc Verhagen (2011). „ISO-Space: The annotation of spatial information in language“. In: *Proc. of the Sixth Joint ISO-ACL SIGSEM Workshop on ISA*, S. 1–9.

Rim, Kyeongmin (Mai 2016). „MAE2: Portable Annotation Tool for General Natural Language Use“. In: *Proceedings of the 12th Joint ACL-ISO Workshop on Interoperable Semantic Annotation* (Portorož, Slovenia).

Spiekermann, Christian, Giuseppe Abrami und Alexander Mehler (Jan. 2018). „VAnnotatoR: a Gesture-driven Annotation Framework for Linguistic and Multimodal Annotation“. In:

Stubbs, Amber (Juli 2011). „MAE and MAI: Lightweight Annotation and Adjudication Tools“. In: *2011 Proceedings of the Linguistic Annotation Workshop V* (Portland, Oregon). Association of Computational Linguistics.

Vincent, L. und P. Soille (1991). „Watersheds in digital spaces: an efficient algorithm based on immersion simulations“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.6, S. 583–598. DOI: 10.1109/34.87344.