Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

AVDT - Automatic Visualization of Descriptive Texts

Christian Spika, Katharina Schwarz, Holger Dammertz, and Hendrik P. A. Lensch

Institute of Media Informatics, Ulm University, Germany

Abstract

Expressing mental images visually as 3D scenes is a time-consuming challenge. Therefore, we employ natural language to facilitate the creation of virtual environments. In this paper, we present a framework, which automatically converts an arbitrary descriptive text into a representative 3D scene. Our system parses a user-written input text, extracts information using techniques from Natural Language Processing (NLP) and identifies relevant units. Based on derived object-to-object relations, our system associates every object with an appropriate 3D model and evaluates spatial dependencies of the entities. The resulting locations are combined based on adequate heuristics in order to create natural looking virtual environments. Finally, a physics engine is used to render a realistic and interactive 3D scene which enables the user to actively manipulate the stage setup.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—I.2.7 [Artificial Intelligence]: Natural Language Processing—

1. Introduction

Realizing thoughts in a 3D scene is still a very time-consuming and difficult process. Typical scene modeling tools tend to be overwhelming at first sight. Before starting to model the scene, the user has to familiarize himself with the supporting graphics software, i.e., learning all the menus and buttons and finding out, how to tweak parameters. After that, the task of actually creating the visualization still remains. Therefore, a language-based approach, in which virtual environments are described and created directly through natural language and which does not rely on special graphics applications, simplifies the process of 3D scene generation.

In this paper, we present a framework called *AVDT - Automatic Visualization of Descriptive Texts*, which automatically generates 3D scenes from natural language. More precisely, an almost arbitrary descriptive text serves as a basis for creating a specific virtual environment. The description consists of the objects occurring in the scene, as well as the spatial relationships between them. Our system concentrates on creating stage setups and, on this account, focuses on the key issues of Information Extraction (IE) as part of NLP, semantics, and graphical representation of a given text.

Fig. 1 shows a high level overview of the AVDT pipeline, based on the two processing elements *Automatic Scene Graph Generation* (Sec. 3) and *Natural Arrangement of Objects* (Sec. 4) each containing several components. First, a descriptive text is parsed and tagged by the IE process (Sec. 3.1.1). This extracted information is then gathered and refined (Sec. 3.1.2) which mainly refers to collecting information about real world objects in the text as well as spa-

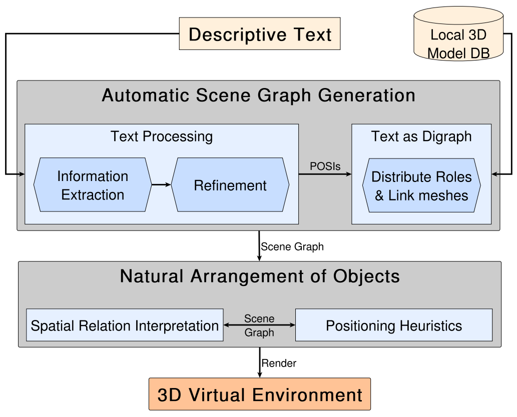


Figure 1: *System overview. A descriptive text is analyzed, extracted information refined and transformed into a directed graph. Then, object-to-object relations are evaluated, combined with positioning heuristics and rendered as 3D scene.*

tial relations that indicate a connection between those entities. Furthermore, additional information (e.g., type, quantity, or position) is added creating new data elements, which we call *Part of Spatial Information* (POSI). A directed graph is then built by filtering unnecessary POSIs out of the text, whereas the remaining ones represent the nodes of the graph (Sec. 3.2). Every POSI representing an object is associated with an appropriate 3D model, which is stored in a local model database. For our processing, we assume that the 3D models we retrieve from Google 3D warehouse[†] are modeled correctly and contain a clearly defined front side. The resulting scene graph builds the basis for the second part of the systems pipeline. The dependencies in the graph and the spatial relations are evaluated by first calculating the location of every POSI in the graph (Sec. 4.1) and, in a second step, applying positioning heuristics (Sec. 4.2) for increasing the "natural" look of the scene. Finally, the resulting virtual environment is rendered by a physics engine into a realistic stage setup enabling the user to fly around or move objects.

2. Related Work

Already quite a few projects investigated the field of natural language input for creating virtual environments. In the following, we present some of the research that is most related to this paper and motivated the development of our system.

The SHRDLU program [Win71] was one of the earliest systems that was able to understand and evaluate natural language. User interaction was allowed via simple english dialogs about a small blocks world shown on an early display screen. SHRDLU was primarily a language parser with the ability to use semantic information and context to interpret natural language input. However, the usable vocabulary for interaction was rather limited and the amount of referenced objects was restricted to a pre-existing environment.

Another system was created by Adorni et al. [ADMF83]. Natural Language Input for Image Generation (NALIG) is able to interpret simple italian phrases of the form *<subject> <preposition> <object> [<reference>]*. The system disambiguates descriptions by defining primitive relationships between objects, represented as taxonomical rules, e.g., *H_SUPPORT(A,B)*.

In 1996, Clay et al. [CW96] implemented the Put system that uses a combination of linguistic commands and direct manipulation to correctly arrange and constrain rigid objects within a virtual scene. Although Put shares the intention of our system to ease the 3D scene creation process, it is limited to pre-existing objects and their spatial arrangements. Besides, it allows only a small subset of english expressions and uses a rigid syntax to formulate placement instructions.

One of the most well-known projects in the field of language-based 3D scene generation is WordsEye, created by B. Coyne and R. Sproat [CS01]. It generates static scenes out of a user-given text. An entered text consists of simple sentences that describe positions of objects and their orientations, colors, textures, and sizes. Although WordsEye realistically visualizes natural language input, the interpretation of spatial relationships often fails and the structure of the input is rather restricted. In order to generate "natural" looking scenes, the user is also required to use parameters for arranging objects, which takes time and effort. Contrary, in our approach we disregard colors and textures because we want to focus on correctly interpreting spatial relations without the need of user interaction, and keep the input more flexible. As WordsEye is closely related to our work, we discuss differences in more detail and show several examples in Sec. 6.

Further, Schwarz et al. [SRC*10] implemented a system which automatically illustrates written natural language with semantically close images they retrieve from online photo collections. Although not aiming at creating 3D scenes, it was the basis for our work due to the linguistic analysis within the visualization process.

[†] http://sketchup.google.com/3dwarehouse/

3. Automatic Scene Graph Generation

This section describes the first part of the AVDT pipeline on which our whole system is based. It consists of analyzing an arbitrary descriptive text and transforming it into a directed graph representation containing all scene relevant data.

3.1. Text Processing

First, basic information about syntax and structure of the text is extracted in a pre-processing step. In order to build a data structure with all relevant information for 3D scene generation, refining of the extracted textual information follows.

3.1.1. Information Extraction using GATE

Forming an important part of Natural Language Processing [JM08], Information Extraction produces structured output from unseen documents [CL96]. Thus, in order to extract grammatical information, our system pre-processes an input text using an open source architecture called GATE (General Architecture for Text Engineering) [CMBT02]. A visualization of the results we use from GATE is shown in Fig. 2.

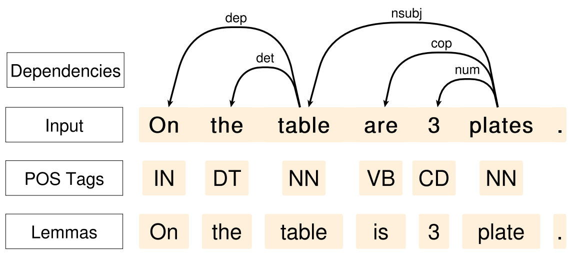


Figure 2: *Visualization of information extracted by GATE.*

POS Tags. Consisting of various NLP software components, GATE includes an IE system annotating each word with its part-of-speech (POS) tag [Hep00]. From those tokens we mainly use nouns (tagged with NN, NNS, NP, or NPS) and prepositions (IN). The nouns form the basis for finding adequate 3D models and the prepositions serve for interpreting spatial relations between objects. Furthermore, cardinal numbers (CD) and coordinating conjunctions (CC) are used.

Lemmas. Additionally, GATE provides a morphological analyzer which adds a token with its lemma form to each word [MRS08]. We apply the lemmatization to nouns only in order to find their singular form. This helps us in identifying an accurate 3D model for our object. A search based on the plural form of a noun would rather result in a whole set of objects. But, as the quantity of an object is calculated separately, the plural form does not serve our system.

Dependencies. In order to derive relationships, our system also uses the Stanford Parser plugin [CMBea10], integrated in GATE. The Stanford Dependencies [dMMM06] represent grammatical relations between words in a sentence and are designed to ease the understanding of those relationships.

3.1.2. Refinement of Textual Information

For the purpose of creating a data structure that stores all the relevant information needed for creating a 3D scene, the next step consists in refining the extracted textual information. This mainly means structuring the input text into interpretable *blocks*, adding useful meta-data concerning quantity and spatial dependencies, and filtering out useless information without spatial contribution. Especially nouns and prepositions are collected for further processing because a preposition can determine a spatial *relation* between two *objects* (nouns). The results of this process are visualized in Fig. 3. Out of the refinement, new data elements arise and we call them *Parts of Spatial Information* (POSI).

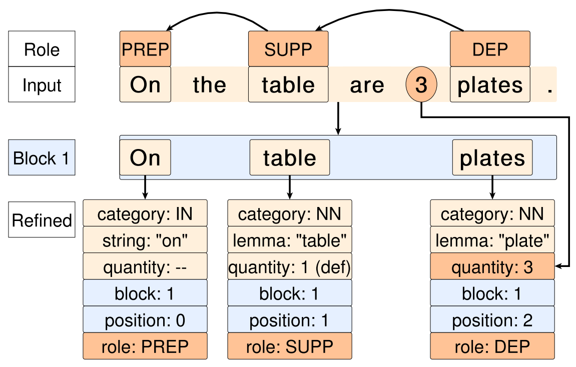


Figure 3: *Collected elements are labelled with their specific POS tag, their lemma, quantity, block index, and their position and role within a block.*

Blocks. For enabling the interpretation process, we segment the text into smaller blocks by splitting sentences at punctuation marks and coordinating conjunctions. Every block consists of a preposition describing a spatial relation and two nouns that embody objects. Thus, it represents an object-to-object relationship. The block index as well as the position of a collected element within the sentence are stored (Fig. 3).

Objects. Real world objects, embodied by nouns, are fundamental for creating a virtual environment out of natural language input. We use the lemma form instead of the normal string. In case of plural nouns, refining also includes adding information about the object quantity. By default, AVDT initializes every detected object with a quantity of 1. If a quantity occurs in front of a noun, the number is replaced.(Fig. 3).

Relations. In grammar, a preposition is a part-of-speech indicating a relation between a noun and other words in a sentence. As we focus on identifying prepositions describing spatial relations in static scenes, we refer to Landau et al. [LJ93] who proposed a list of spatial prepositions in English. In order to ease the interpretation process (Sec. 4.1), we collected the prepositions describing static relations, grouped them according to their semantic meaning, and chose an arbitrary representative out of each class. *Under*, *underneath*, *below*, *beneath*, e.g., form a group with the representative *under*. Due to very small and insignificant differences in the meaning of the spatial prepositions within a group, we interpret them similar to the representative one. Furthermore, some spatial relations are hidden in compounds of a preposition and a prepositional phrase, e.g., "*on the left side*". In general, such a part of a sentence is used with another prepositional phrase construction like "*of the table*". For achieving a homogeneous spatial-relation-evaluation system, AVDT identifies spatial components like "*left*" and treats them as normal prepositions (Fig. 4).

Input:	On	the	left	side
POS Tags:	IN	DT	NN	NN
Output:			left	
Retag:			IN	

Figure 4: *Prepositional phrases are identified and spatial components, such as* "left"*, are extracted and refined.*

Roles. In order to indicate the purpose of an object within a block, it gets associated with a specific tag which is defined based on the sentence structure. Therefore, we use the dependencies extracted from GATE as a basis. Whereas dependency parsers link heads with dependents [Cov01], we derive a structure where the head object is directly connected with its dependent object (Fig. 3). From this we retrieve the two roles of a *supporter* and a *dependent*. A supporter is specified as the noun following a preposition and not being related to another noun. Subsequently, a dependent refers to a noun that is grammatically related to a supporter.

Ordering. Saving a position information for the collected elements within a sentence is important as their chronological order within a block is not always the same. In order to keep our system as flexible as possible concerning the input technique, a user is allowed to enter his text in various ways. For example "*On the table is a vase.*" is accepted as well as “*A vase is on the table.*”. Because AVDT interprets each block as a sequence with the following order,

$$\text{Preposition} \rightarrow \text{Supporter} \rightarrow \text{Dependent}$$

the system checks the elements of every incoming block for their correct arrangement. Therefore, sentences like “*A vase is on the table.*” are automatically reordered which leads to a robust and less sensitive input interpretation.

The resulting meta-data enriched POSIs can now be used to build a directed graph representation.

3.2. Representing Text as a Directed Graph

The illustration in form of a directed graph eases the process of calculating spatial relations because it clearly presents the dependencies between the retrieved POSIs and is simple to traverse. The noun POSIs, either acting as a supporter or a dependent in their blocks, are processed into the nodes of a directed graph. Furthermore, we identify nouns that refer to the same object and link them to the same 3D model.

Graph Structure. While the root represents the origin of the scene, i.e., a root element is the only independent POSI in the graph, the leaf level contains all dependents that do not support any other object. Consequently, any other POSI occurring in the graph represents an object that supports another element but is also a dependent at the same time.

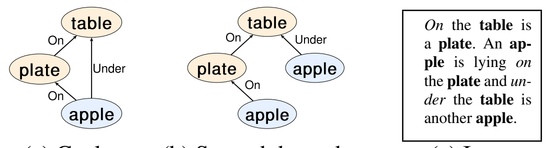
Link with 3D Meshes. As nouns within a descriptive text often refer to the same real world objects, we invented several atomic rules (Table 1) in order to ascertain that POSIs that refer to the same object are linked to the same 3D object. We also link POSIs that use the same noun but refer to a different object in the real world with a new, separate copy of the same 3D model. Considering the sample text “*On the table is a plate. Beside the plate is a spoon.*”. Obviously, both sentence blocks include the noun POSI “*plate*”. Whereas in block 1 the plate implies the role of the dependent, it appears as a supporting POSI in block 2. Our rules for solving linguistic ambiguities define an incoming POSI, that has to be processed, as $POSI_i$. Already evaluated supporter or dependent POSIs are marked as either $POSI_{Sup}$ or $POSI_{Dep}$. Since the POSI-related block plays an important role, it is illustrated by *B(value)*, where the value is either an incoming, supporting, or dependent POSI. Furthermore, the word *cycle* in rule 6 refers to a linguistic cycle (Fig. 5).

All rules compare the lemma form of an incoming POSI with the lemma form of the existing POSI. Therefore, the “IF” clauses are true only if the lemmas of the two POSIs are the same. Rule 1 links a new incoming and not yet existing POSI with a new mesh. Rules 2 and 3 consider repeated supporters or dependents. We thereby assume that a supporter may serve for various dependents (2), whereas various dependents may depend from different supporters (3). Rules 4 and 5 resolve textual dependencies. This considers the fact that a POSI may act as a supporter as well as a dependent at the same time. Although such POSIs, containing different roles (as a supporter and a dependent POSI), have to point

```
1) IF !( $POSI_i$.exists )
      LINK $POSI_i$ WITH new mesh

2) IF ( $POSI_{i_{Sup}}$.lemma == $POSI_{Sup}$.lemma )
      LINK $POSI_{i_{Sup}}$ WITH $POSI_{Sup}$.mesh

3) IF ( !( B( $POSI_{i_{Dep}}$ ) == B( $POSI_{Dep}$ ))
      && ( $POSI_{i_{Dep}}$.lemma == $POSI_{Dep}$.lemma ))
      LINK $POSI_{i_{Dep}}$ WITH new mesh
   ELSE ABORT BECAUSE Repetition

4) IF ( $POSI_{i_{Sup}}$.lemma == $POSI_{Dep}$.lemma )
      LINK $POSI_{i_{Sup}}$ WITH $POSI_{Dep}$.mesh

5) IF (( B( $POSI_{i_{Dep}}$ ) FOLLOWS B( $POSI_{Sup}$ ))
      && ( $POSI_{i_{Dep}}$.lemma == $POSI_{Sup}$.lemma ))
      LINK $POSI_{i_{Dep}}$ WITH $POSI_{Sup}$.mesh
      LINK $POSI_{Sup}$ WITH
               ( $POSI_{Sup}$ FROM $POSI_{i_{Dep}}$ )
   ELSE LINK $POSI_{i_{Dep}}$ WITH new mesh

6) IF ( cycle )
      LINK $POSI_{i_{Dep}}$ WITH new mesh
```

Table 1: *Atomic rules link incoming* $POSI_i$ *either with mesh of an existing* POSI *(same lemma) or with a new 3D model.*

to the same mesh, it is important that they keep their information about the spatial relation (the supporting POSI of the former dependent is saved). As can be seen in rule 4, POSIs that act as supporter and dependent at the same time are created by linking the dependent POSI to the 3D model of the supporter POSI. Furthermore, the supporter POSI receives information about the supporter of the dependent POSI. This way, both POSIs are clustered under the same mesh by keeping all relevant dependency information at the same time.



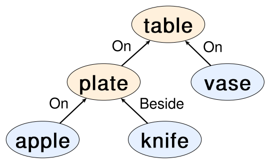
On the **table** is a **plate**. An **apple** is lying *on* the **plate** and *under* the **table** is another **apple**.

(a) Cycle (b) Second dependent (c) Input

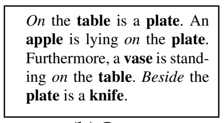
Figure 5: *By creating an additional dependency, the linguistic cycle is dissolved and a directed graph is achieved.*

Finally, rule 6 is responsible for linguistic cycles which may be used in order to create a stack of different objects. In this case, we want all those objects to obtain their own 3D model. The example in Fig. 5 shows that a cycle is dissolved by creating an additional dependency for achieving a directed graph structure. We create a second dependent POSI for the “apple” and link it to the table. This results in a clean hierarchical structure that is easy to be evaluated afterwards.

Clean Representation. By deleting remaining duplicates, we achieve a clean hierarchical representation of the analyzed text that can be seen as a directed graph with one root. An example is visualized in Fig. 6, where the AVDT system determines that the POSI, representing the table, is the root of the graph, since it is the only POSI that does not depend on another one. Consequently, the vase and plate POSIs represent dependents that also function as supporters. The remaining leafs of the graph illustrate dependents.



(a) Clean representation



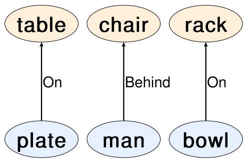
On the **table** is a **plate**. An **apple** is lying *on* the **plate**. Furthermore, a **vase** is standing *on* the **table**. *Beside* the **plate** is a **knife**.

(b) Input

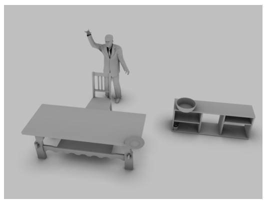
Figure 6: *After duplicates are deleted, a clean directed graph representation is achieved.*

Unconnected Text. Another aspect refers to the problem of unconnected text descriptions. In case of incoherent textual information, our system is able to generate multiple digraphs that represent such texts. Multiple graphs are dissolved by placing an evaluated graph at a prior one (Fig. 7).

On the **table** is a **plate**. The **man** is standing *behind* the **chair**. *On* the **rack** is a **bowl.**



(a) Multiple graphs



(b) Resulting scene

Figure 7: *Unconnected text descriptions (input on top) create multiple independent graphs. Every graph is evaluated separately and situated at a prior calculated graph.*

Representing a given arbitrary text as a directed graph significantly eases the process of calculating spatial relations between two related POSIs. For instance, in order to calculate the final position of the apple in the example mentioned above (Fig. 6), one must only regard the position of the supporting plate POSI. Due to the hierarchical structure of the graph, the plate already contains its final position and, therefore, the location of the apple can easily be computed as described in the following section.

4. Natural Arrangement of Objects

As we focus on analyzing grammatical syntax and dependencies in natural language and correctly mapping the results to objects and their relations within static scenes, any further parameters associated with noun POSIs, except information about quantity, are ignored. Aiming at enabling the user to easily depict a scene while receiving a basic nice looking result, the interpretation of spatial relations should not only be correct and accurate. Rather, it should also be capable of delivering a virtual scene with "natural" arranged objects, i.e., like one would expect it in the real world.

4.1. Spatial Dependencies Interpretation

As already mentioned in Sec. 3.1.2, we group prepositions concerning static spatial relations and chose a representative for each class. The representatives are all interpreted uniquely. Our system creates an axis-aligned bounding box (BB) for each object endorsed in the graph. Next, AVDT calculates the position of a bounding box, depending on the saved spatial relation and the final position of the supporter saved within the currently viewed POSI. The final location is then stored in a transformation matrix, which is combined with a rigid body later on. Finally, all rigid bodies are processed by a physics engine and the final scene is rendered.

The pseudo-code shown in Table 2 illustrates how the spatial relation *beside* is calculated. Every dependent is placed randomly at one of the four sides of the bounding box of its supporter. Some more examples of relations we im-

```
FOR ( random side of supporters BB )
   CALCULATE height position for dependent
   IF ( random side not yet used )
      PLACE dependent
   ELSE IF ( other side free )
      PLACE dependent
   ELSE
      PREVENT collision PLACE dependent
```

Table 2: *Pseudo-code of preposition "beside"*

plemented and their resulting renderings are visualized in Fig. 8. The spatial relation *in*, for example, is evaluated by placing the dependent on the bottom of the supporters model. Because an object does not stand exactly on the center within another entity, random coordinates are used in order to vary the position of the dependent. If our system detects the spatial dependency *on*, the dependent is placed on top of the supporters bounding box. The position on the surface is not fixed and the system calculates random coordinates over the top surface of the supporter. However, in order to prevent a dependent from floating in the air or falling down when physics are activated, the computed coordinates do not cover the entire surface of the bounding box. Further, the spatial relations *left*, *right*, *behind*, and *front* are interpreted as fixed on a specific side of the bounding box. Our system evaluates the relationship *above* by applying no contact between the dependent and the supporter. Therefore, the depending object is placed in the "air" above the center of the supporter. Subsequently, the spatial preposition *under* is processed by situating the dependent below the supporter. This also requires changing the position of the supporting object as well as altering the locations of other possibly participating entities. However, the correlation *around* differs as being interpreted as a uniform distribution of one or more dependents around their supporter, placed on a circular orbit. This example especially illustrates that AVDT easily can be extended in order to increase the positioning possibilities.

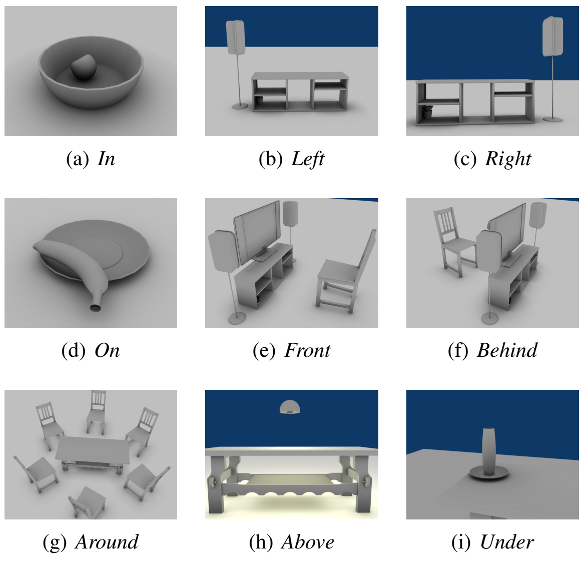


Figure 8: *Renderings for some object-to-object relations.*

4.2. Positioning Heuristics for Natural Appearance

For improving the "natural" look of a 3D virtual environment, heuristics are applied within AVDT. These rough rules are used while a spatial relation is evaluated and calculated.

Distance Heuristic. The first heuristic refers to the distance of dependents to their supporter and calculates the distance ratio based on:

$$\underset{dim_{Dep} > dim_{Sup}}{H_{dist}} = \frac{(dim_{Dep} - dim_{Sup})}{c_d}$$

This means that in case the dependent has a bigger dimension (dim_{Dep}) than its supporting object (dim_{Sup}), the distance is evaluated from the difference of both dimensions. The result is divided by a constant c_d for normalizing the calculated distance to a realistic one. The larger the constant is chosen, the smaller the space between a dependent and its supporter gets. Whereas the maximum distance is achieved for $c_d = 1$, a value of $c_d = 1.7$ seems to create realistic stage setups. Based on this heuristic, a bed which is placed near a table is likely to be further away than a chair (Fig. 9).

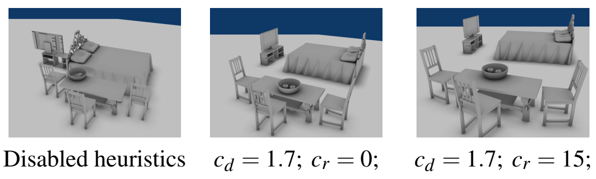


Figure 9: *Visualizations for varied positioning heuristic parameters show different looks above same stage setup.*

Rotation Heuristic. For further increasing the realism of a 3D scenario, we use a second positioning heuristic, which is used to ensure that every dependent is facing its supporter in the scene. For example, an armchair that is standing on the left side of a table should face the supporting table that is located on its right side. Moreover, this heuristic applies a little random rotation to all of the occurring objects within a scene in order to achieve an "untidy" appearance. The rotation heuristic is defined as:

$$H_{rot} = \cos(\alpha + (\xi * c_r)) + \sin(\alpha + (\xi * c_r)) * \vec{V_r}$$

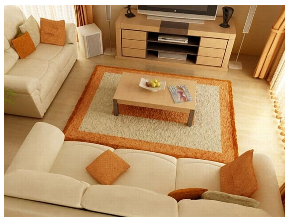
AVDT uses rotation quaternions, which are applied to the transforms of the axis-aligned bounding boxes. By passing an appropriate angle α as well as a vector describing the rotation axis, AVDT adjusts a dependent on its supporter. Further, a random number $\xi \in [-1, 1]$ combined with a rotation constant $c_r \in [0, 360]$ is used to destroy the perfect alignment of a dependent to its supporter and results in an untidy scene. In general, one would never arrange one's furniture such that it is perfectly aligned in the room ($c_r = 0$). The same principle applies in AVDT. Every object is randomly rotated in 2D space (3D for preposition *in*) in order to increase the naturalness within a virtual environment (Fig. 9).

By applying positioning heuristics on the results of the position calculation, our system is able to further increase the naturalness within a created virtual environment, without using any extra parameters, e.g., size or distance specifications. Also, by varying the different parameters of the heuristics, one can achieve very different results for the same input text.

5. Results

In the following, we present some results for different input texts. As previously described, every POSI has been assigned with an adequate 3D model and rigid body. Also, every spatial relation has been evaluated and combined with positioning heuristics. Finally, the resulting virtual environment is rendered interactively, offering the user the opportunity to fly around and manipulate objects within the scene.

A **table** is standing *on* the **carpet**. On the *left* side of the **carpet** is a **sofa**. A **couch** is in *front* of the **carpet** and *behind* the **carpet** is a **shelving**. The **TV** is placed *on* the **shelving**. On the *left* side of the **shelving** is a **loudspeaker**. A **subwoofer** is on the *left* side of the **loudspeaker**. Another **loudspeaker** is on the *right* side of the **shelving**. On the *left* side of the **TV** is a **statue**. *On* the **table** stands a **bowl** and *in* the **bowl** lie **2 apples**.



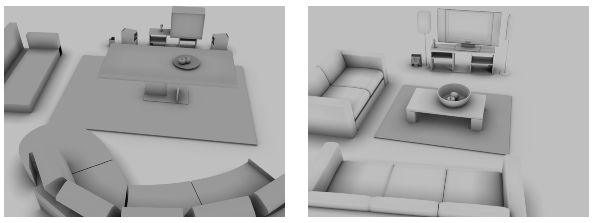


Figure 10: *Top: Input text (left) describing common living room photo (right). Bottom: Resulting indoor visualizations by AVDT using different models.*

We compare an example of indoor scene visualizations (Fig. 10) of AVDT to an outdoor scene result (Fig. 11). The indoor examples show the same scene with different models and are based on a simple text describing a photo of a common living room. As can be seen, the various objects are nicely placed in the space and are correctly arranged according their spatial dependencies. Besides, the adapted models (bottom right) result in a quite satisfying interpretation of the real world picture (top right). As the capabilities of the AVDT system are not limited to the generation of indoor scenes, the proposed algorithms and mechanisms are also able to create natural looking landscape scenarios. The visualization shown in Fig. 11 gives a nice example of an outdoor scene rendering in which the arrangement of the several objects leads to a realistic impression.

In *front* of a **cottage** is a **tree**. On the *left* side of the **cottage** are 2 **trees** and 3 **trees** are growing on the *right* side of the **cottage**. *Behind* the **cottage** is another **tree**. A **bench** is standing on the *left* side of the first **tree** and in *front* of the **bench** is a **man**. In *front* of the first **tree** is a **fence**. An **oldtimer** waits in *front* of the **fence**. On the *left* side of the **oldtimer** is a **lantern** and another **lantern** is on the *right* side of the **oldtimer**.

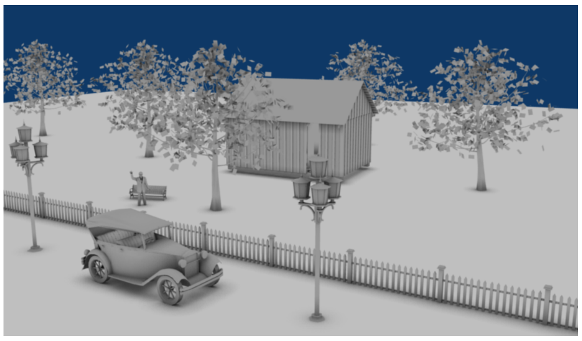


Figure 11: *AVDT illustration of an outdoor scenario (bottom) based on an descriptive input text (top).*

6. Discussion and Future Work

After the comparision to WordsEye, some further limitations are discussed serving as a basis for future work at the same time. Although our work already creates nicely looking 3D scenes from natural language, several research areas could increase the capabilities as well as the usability of AVDT.

Comparision to WordsEye. As already mentioned is WordsEye [CS01] one of the most well-known projects in the field of language-based 3D scene generation. Natural language is visualized in a nice and realistic way. Thus, we want to compare some main differences concerning our work. In order to ease the modeling process, we mainly focus on an automatic realistic placement of objects. Thus, we treat every class of object-to-object relations differently and interpret spatial relations correctly, whereas WordsEye often needs additional user interaction and parameter tuning for realistic spatial arrangements. Some examples for failing placements in WordsEye are shown in Fig. 12 whereas our correct ones were already mentioned in Fig. 8. Besides, contrary to WordsEye, AVDT is capable of dealing with linguistic cycles. Furthermore, whereas WordsEye requires a defined pattern for user input, our system allows more comfortable input because of syntax reordering. An example is given in Fig. 13. Although our system allows for more flexible linguistic input, WordsEye, on the other hand, provides colors, textures, etc. As those attributes add a more realistic appearance to a scene, a further step in AVDT will consist in interpreting and including more context such as adjectives, and adverbs.

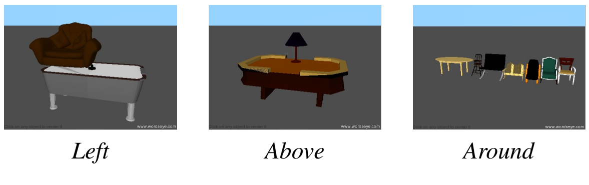


Figure 12: *Failing spatial interpretations in WordsEye.*

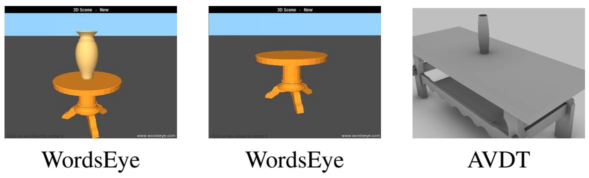
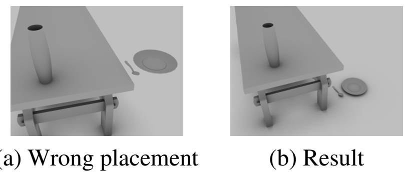


Figure 13: *Comparison to WordsEye concerning syntax stability. Input (a):* “The vase is on the table.” *Input (b):* “On the table is a vase.” *Both input texts result in (c) by AVDT.*

Positioning. Although the positioning system is stable, misplacements of objects may still occur (Fig. 14). This can be solved by developing a method that automatically adjusts the position of each object by traversing the graph backwards. At this point, a user can solve this problem by iterating the scene once again or by manually repositioning the objects. Further, wrong placements appearing due to limitations of a bounding box can be avoided by implementing object-tight bounding boxes or switching to triangle-based collision detection. Unfortunately, this would increase computation time. Besides, as described in Sec. 4.2, AVDT does not evaluate any extra parameters for the purpose of creating natural looking virtual environments. Therefore, the AVDT system cannot evaluate spatial relations like “*On the second plate is a spoon.*” Furthermore, although our system already improves the appearance of a created 3D scene by applying positioning heuristics, it has no knowledge about how objects are used in real life, which complicates the determination of the orientation of objects and might also lead to



(a) Wrong placement (b) Result

A **vase** is standing *on* the **table**. *Beside* the **vase** is a **plate**. A **spoon** is *left* to the **plate**.

(c) Input

Figure 14: *Lack of collision tests may lead to wrongly placed objects (a). At this stage, the misplaced objects fall to the ground due to the underlying physics engine (b).*

wrongly interpreted locations. By combining the used NLP techniques with common-sense knowledge, as introduced in ConceptNet [Hav07], the positioning of objects could be significantly eased and enhanced.

Text Processing. Concerning text, several passages are filtered out intentionally. For example, our system does not allow repetitions since they are generally not used to describe a scene, at least in the normal linguistic usage. Moreover, AVDT evaluates cyclic expressions like "*On the table is a vase. A flower is in the vase. The flower is under the table.*" by generating a second dependent object for the flower. Both cases, repetitions and cyclic text segments, are recognized and filtered out or corrected by the rules described in Sec. 3.2. Anyway, due to the very complex nature of natural language, our system is not capable of handling all kinds of different text. Further enhancements in the amount of natural language processing could be achieved by extending the information extraction process. Using linguistic databases like WordNet [Mil95] for enlarging the semantic analysis of a text, or WordNet-Affect [Val04] for filtering unnecessary nouns like feelings or emotions, could improve the natural language understanding of AVDT.

Natural Appearance. Finally, for further raising the naturalness of a scene, the interpretation of adverbs or adjectives that indicate attributes of objects could be added. Besides, it is not only important to position and orient objects realistically, but also to visualize their "behaviour". This includes depicting poses for (humanoid) objects by evaluating verbs and/or common-sense knowledge, or using the incorporated physics engine for physical simulations, e.g., skeletal dynamics, fluids, or surfaces.

7. Conclusion

The work we presented in this paper further extends the border of language-based 3D scene generation. Our AVDT system enables a user to quickly generate virtual environments by using natural language as input. Starting from a descriptive text, relevant information about objects and spatial relations are gathered and refined. The findings are used to link retrieved entities to appropriate 3D models as well as deriving a directed graph representation of the text. With the aid of that digraph, spatial relations between objects are evaluated. The resulting locations and models are finally assembled in an interactive virtual environment.

Especially the development of a rule-based text interpretation module, as well as the clear and easy to traverse hierarchical graph representation and, also, the accurate interpretation of spatial relations in combination with positioning heuristics improved the success of creating virtual environments close to a user-given input text. Several results illustrated how often the intention of an underlying descriptive text is already properly visualized by the AVDT system. This work can be further extended and one can imagine numerous applications in which there is need for transferring spatial ideas in visual communication.

References

[ADMF83] ADORNI G., DI MANZO M., FERRARI G.: Natural language input for scene generation. In *Proceedings of the first conference on European chapter of the Association for Computational Linguistics* (1983), Association for Computational Linguistics, pp. 175–182. 2

[CL96] COWIE J., LEHNERT W.: Information Extraction. *Communications of the ACM 39* (1996), 80–91. 2

[CMBea10] CUNNINGHAM H., MAYNARD D., BONTCHEVA K., ET AL.: *Delevoping Language Processing Components with GATE Version 6 (a User Guide)*, version 6 ed. The University of Sheffield, November 2010. 3

[CMBT02] CUNNINGHAM H., MAYNARD D., BONTCHEVA K., TABLAN V.: A framework and graphical development environment for robust NLP tools and applications. In *ACL - The Association for Computational Linguistics* (2002), pp. 168–175. 2

[Cov01] COVINGTON M. A.: A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference* (2001), pp. 95–102. 3

[CS01] COYNE B., SPROAT R.: WordsEye: An Automatic Text-to-Scene Conversion System. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), Annual Conference Series, ACM Press / ACM SIGGRAPH, pp. 487–496. 2, 7

[CW96] CLAY S. R., WILHELMS J.: Put: Language-Based Interactive Manipulation of Objects. *IEEE Computer Graphics and Applications 16*, 2 (Mar. 1996), 31–39. 2

[dMMM06] DE MARNEFFE M.-C., MACCARTNEY B., MANNING C. D.: Generating typed dependency parses from phrase structure parses. In *LREC* (2006). 3

[Hav07] HAVASI C.: ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *the 22nd Conference on Artificial Intelligence* (2007). 8

[Hep00] HEPPLE M.: Independence and commitment: assumptions for rapid training and execution of rule-based POS taggers. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics* (Morristown, NJ, USA, 2000), ACL '00, Association for Computational Linguistics, pp. 278–277. 2

[JM08] JURAFSKY D., MARTIN J. H.: *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition (Second Edition)*. Prentice Hall, 2008. 2

[LJ93] LANDAU B., JACKENDOFF R.: "What" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences 16(2)* (1993), 217–238. 3

[Mil95] MILLER G. A.: Wordnet: A lexical database for english. *Commun. ACM 38* (November 1995), 39–41. 8

[MRS08] MANNING C. D., RAGHAVAN P., SCHÜTZE H.: *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 3

[SRC*10] SCHWARZ K., ROJTBERG P., CASPAR J., GUREVYCH I., GOESELE M., LENSCH H. P. A.: Text-to-Video: Story Illustration from Online Photo Collections. In *Knowledge-Based and Intelligent Information and Engineering Systems* (2010), vol. 6279 of *Lecture Notes in Computer Science*, Springer, pp. 402–409. 2

[Val04] VALITUTTI R.: WordNet-Affect: an Affective Extension of WordNet. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation* (2004), pp. 1083–1086. 8

[Win71] WINOGRAD T.: *Procedures as representation for data in a computer program for understanding natural language*. Tech. Rep. MAC AI-TR-84, MIT, Cambridge, 1971. 2