

Annotation Tools and Knowledge Representation for a Text-To-Scene System

Bob Coyne¹ Alex Klapheke¹
Masoud Rouhizadeh² Richard Sproat³ Daniel Bauer¹

(1) Columbia University

(2) Oregon Health Sciences University

(3) Google

coyne@cs.columbia.edu, agk2118@columbia.edu, masoud@cslu.ogi.edu,
rws@xoba.com, bauer@cs.columbia.edu

ABSTRACT

Text-to-scene conversion requires knowledge about how actions and locations are expressed in language and realized in the world. To provide this knowledge, we are creating a lexical resource (*VigNet*) that extends FrameNet by creating a set of intermediate frames (*vignettes*) that bridge between the *high-level* semantics of FrameNet frames and a new set of *low-level* primitive graphical frames. Vignettes can be thought of as a link between *function* and *form* – between what a scene means and what it looks like. In this paper, we describe the set of primitive graphical frames and the functional properties of 3D objects (*affordances*) we use in this decomposition. We examine the methods and tools we have developed to populate VigNet with a large number of action and location vignettes.

KEYWORDS: text-to-scene conversion, world knowledge, frame semantics, visual semantics, linguistic annotation, crowdsourcing.



Figure 1: Mocked-up scenes using the WASH-FRUIT-IN-SINK vignette (“John washes the apple”) and WASH-FLOOR-W-SPONGE vignette (“John washes the floor”).

1 Introduction

3D graphics authoring is a difficult process, requiring users to master a series of complex menus, dialog boxes, and often tedious direct manipulation techniques. Natural language offers an interface that is intuitive and immediately accessible to anyone, without requiring any special skill or training. The WordsEye system (Coyne and Sproat, 2001) lets users create 3D scenes by describing them in language. It has been used by several thousand users to create over 10,000 scenes by merely describing them. We have tested WordsEye as an educational tool with rising 5th grade children in a summer enrichment program where it was found to significantly improve literacy skills over the students who had taken the more traditional version of the course (Coyne et al., 2011b). As one of the students said “When you read a book, you don’t get any pictures. WordsEye helps you create your own pictures, so you can picture in your mind what happens in the story.” The students were also introduced to WordsEye’s face and emotion manipulation capabilities – the children loved including themselves and other people in scenes and modifying the facial expressions. We are currently experimenting with automatically depicting Twitter tweets as a way to bring text-to-scene visualization to a wider audience and to test the limits of the system in an open domain with ill-formed text. In this paper we describe a new set annotation tools and the graphical primitives we have developed in order to build a knowledge base that maps linguistic constructs into semantic frames representing spatial relations and other graphical relations.

WordsEye currently focuses on directly expressed spatial relations and other graphically realizable properties. As a result, users must describe scenes in somewhat stilted language. Our goal is to build a comprehensive text-to-scene system that can handle a wide range of input text. When considering sentences such as *John is washing an apple* and *John is washing the floor*, it becomes apparent that different graphical knowledge is needed to generate scenes representing the meaning of these two sentences (see Figure 1): the human actor is assuming different poses, he is interacting differently with the thing being washed, and the water, present in both scenes, is supplied differently. If we consider the types of knowledge needed for scene generation, we find that we cannot simply associate a single set of knowledge with the English verb *wash*. Instead we need to take into account the arguments of the verb as well.

Knowledge about verbs and their arguments is encoded in FrameNet (Ruppenhofer et al., 2010). FrameNet is a lexical resource focused on representing semantic relations and their possible instantiations in lexical items. In FrameNet, lexical items are grouped into frames that represent their shared semantic structure. A FrameNet frame consists of a set of frame-based roles, called *frame elements* (FEs). For example, the `COMMERCE_SELL` frame includes frame elements for Seller, Goods, and Buyer. These and other FEs represent the key roles that characterize the

meaning of the lexical units in that frame. Frames can contain any number of individual lexical units. The `COMMERCE_SELL` frame, for example, has lexical units for words like *retail*, *sell*, and *vend*. The exact expression of FEs for a given sentence constitutes what FrameNet refers to as a *valence pattern*. Valence patterns map grammatical roles to frame element for a given verb. Every verb typically has many valence patterns, representing the various ways that verb can be used in sentences. So, for the verb *give*, the sentence *John gave the book to Mary* has the valence pattern of: ((Donor Subject) (Theme Object) (Recipient Dep/to)). And *John gave Mary the book* has the valence pattern of ((Donor Subject) (Recipient Object) (Theme Dep/NP)). FrameNet also supports *frame-to-frame relations* – these allow frames to be inherited, to perspectivize each other, or to map to a sequence of temporally ordered subframes.

(Coyne et al., 2011a) describes an extension to FrameNet called *vignettes*. Vignettes are frames that represent the graphical realization of actions and compound objects such as locations. Vignettes can be thought of a bridge between the *high-level* semantics encoded by FrameNet and the *low-level* semantics (spatial relations and other graphical properties of objects) required to construct a 3D scene. Vignettes inherit high-level semantics from FrameNet via normal frame-to-frame inheritance and decompose into low-level graphical frames using a new `SUBFRAME-PARALLEL` frame-to-frame relation. Vignettes can be defined not only for actions but also for locations or any other compound objects. For example, a living room might contain a sofa, coffee table, and fireplace in a particular arrangement

There is a long history in artificial intelligence and cognitive linguistics of decomposing meaning into semantic primitives. These efforts fall into two broad classes – those focusing on primitive features of objects used to distinguish one object from another (for example in prototype theory (Rosch, 1973)) and those focused on state changes, temporal relations, and causality (Miller and Johnson-Laird, 1976). Conceptual Dependency (Schank and Abelson, 1977) is an early representational system that specifies a small number of state-change primitives into which all meaning is reduced. In lexical conceptual structure (Jackendoff, 1985), lexical relations are decomposed into a similar set of primitives. VerbNet (Kipper et al., 2000) grounds verb semantics into a small number of causal primitives representing temporal constraints tied to causality and state changes. In contrast to these causally and temporally oriented approaches, vignettes map semantics into sets of graphical constraints active at a single moment in time. This allows for and emphasizes *contextual entailment* rather than causal and temporal reasoning.

In order to apply graphical primitives to objects, it is necessary to know how objects are used and can be manipulated. The concept of *affordances* (Gibson, 1977; Norman, 1988) was introduced in the study of ergonomics and the psychological interpretation of the environment and has been examined as well from a philosophical perspective (Haugeland, 1995). Affordances are traditionally considered to be the qualities of objects in the environment that allow people or animals to interact with them. For example, the door of a refrigerator provides an affordance that allows access to the interior of the refrigerator, and the handle on a door provides an affordance that allows the door to be grasped in order to open and close it. We take a slightly broader view and include as affordances any functional or physical property of an object that allows it to participate not only in actions with people but also in relations with other objects.

One of our main tasks is to define vignettes to represent a wide variety of actions and locations. Previous work explored different methods for building location vignettes. Sproat (2001) attempted to extract associations between actions and locations from text corpora. While producing interesting associations, the extracted data was fairly noisy and required hand

editing. Furthermore, much of the information that we are looking for is common-sense knowledge that is taken for granted by human beings and is not explicitly stated in corpora. In other work Rouhizadeh et al. (2010) and Rouhizadeh et al. (2011b), Amazon Mechanical Turk (AMT) was used to collect information about locations of different objects, their parts, and surrounding objects. Various corpus association and WordNet similarity measures were applied to filter the undesirable AMT inputs. Reasonably clean data was achieved by this approach. In Rouhizadeh et al. (2011c) AMT was used for building a low-level description corpus for locations by collecting free-form text descriptions of room locations based on their pictures. The WordsEye NLP module was used to extract location elements from processed descriptions. Objects and other elements of locations were extracted in the form of `RELATION-GROUND-FIGURE`. Directly collecting objects of locations with AMT was investigated in Rouhizadeh et al. (2011a). AMT was then used for annotating the orientation of those objects (for more details see subsection 4.1).

This paper is structured as follows. In Section 2, we describe the graphical primitives used in vignettes as well as the set of affordances on 3D objects that enable vignettes to specify their arguments in a generic way. In Section 3, we describe how we choose and process lexical patterns to serve as input valence patterns to be used in defining action vignettes. In Section 4, we describe the annotation tools and graphical user interfaces we have developed to define location vignettes and action vignettes (with their associated valence patterns). We also describe the user interface we developed for assigning affordances and normalized part names to 3D objects. We conclude and describe future work in Section 5.

2 Vignettes and Affordances

In this section we examine, in more detail, the set of primitive graphical frames used by vignettes and the 3D object affordances used in applying those graphical primitives.

2.1 Primitive Graphical Frames

We observe that visual scenes can be decomposed into a relatively small and recurring set of primitive graphical relations. These relations represent different spatial and graphical properties such as positions, orientations, sizes, surface properties, character poses, and facial expressions. So, for example, *the man washing the floor* can be decomposed into a set of relations consisting of the man in a kneeling pose on the floor, with a bucket to his side, and holding a sponge that he applies to the floor. To capture the different manners of performing actions within the same high-level semantic frame, we define a specialized type of sub-frame called a *vignette* (Coyne et al., 2011a). Vignettes can be thought of as a bridge between *form* (the way scenes look) and *function* (what is happening or conveyed in a scene). They encode and map low-level graphical relations to the high-level semantics encoded by FrameNet. To make this decomposition, the following sets of primitive graphical frames (grouped by type) are used:

Venue and time of day: This specifies the typical time of day and the *venue*. The time of day is graphically realized by controlling the position and brightness of light sources within the scene (such as the sun position). The venue represents the local human-scale area where the action takes place and what can be seen at one time. For example, a living room or a kitchen would be a venue, but a typical house as a whole would contain many venues. The venues, themselves, are arrangements of 3D objects, and hence can be represented by location vignettes.

Holding/Touching target or patient: These specify that an agent is holding or touching an object. They vary in how the target object is situated. Arguments are provided to specify the particular body part and pose involved in holding or touching the object.

GRASP/TOUCH-ON-SELF: *button one's shirt*

GRASP/TOUCH FIXTURE: *hold a doorknob*

GRASP/TOUCH NON-FIXTURE: *hold a coffee mug*

POSITION-THEME-MANUALLY: *put a picture on a wall*

Apply Handheld Instruments: These specify a handheld instrument being applied to an object. Handheld instruments typically have affordances (see section 2.2), such as a HANDLE that allow them to be held and a INSTRUMENT-HOTSPOT, such as the blade of a knife or tip of a pencil, that is applied to a target. Undirected instruments (playing a violin) often involve specialized poses that are associated with the use of those instruments.

APPLY-INSTRUMENT-TO-HELD-PATIENT: *write on a handheld notepad*

APPLY-INSTRUMENT-USING-WORKSURFACE: *cut carrots on a table*

APPLY-INSTRUMENT-TO-TARGET: *paint the wall*

POSITION-THEME-WITH-INSTRUMENT: *roast the marshmallow*

USE-UNDIRECTED-INSTRUMENT: *play the violin. talk on the phone*

Using stationary machines/fixtures: *Fixtures* are large stationary objects. *Machines* are fixtures that can be operated to achieve some effect. Machines, such as ovens, can affect a PATIENT. The patient can rest on the machine's PATIENT-AREA affordance. Other machines, such as vending machines, can have INPUT/OUTPUT-AREA affordances. They differ from instruments in that they are not held or otherwise supported by the user. See Section 2.2 for a listing of affordances.

APPLY-MACHINE-TO-PATIENT: *boil the potatoes on the stove*

USE-FIXTURE/MACHINE: *open the door*

Looking/gesturing at target: In these frames, the agent is looking or gesturing at an object or in some direction. The particular pose or manner of gesturing can be specified.

LOOK-AT-TARGET: *glance across the room*

GESTURE-AT-TARGET: *wave at the stranger*

LOOK-AT-WITH-INSTRUMENT: *take a picture of friends*

Aiming and projectiles: In these frames, the agent is aiming or hurling an object at a remote target. A projectile can be either included or not.

HURL-PROJECTILE-AT-TARGET: *throw the stone, kick the ball*

AIM-INSTRUMENT-AT-TARGET : *shoot the rifle*

AIM-FIXTURE/MACHINE-AT-TARGET : *shoot the cannon*

Agent-in-Motion: In these frames, the agent is moving, possibly in or with a vehicle. This frame includes a source or goal location and an area or path for the motion.

SELF-MOTION: *swim across the lake*

USE-VEHICLE: *ride the horse*

PUSH-OR-PULL-FIXTURE/VEHICLE: *push the wheelbarrow*

Humans and Poses: These include facial expressions, body poses, and other body states. In the two-person interaction case, a set of specialized two-person poses can be specified.

STANCE: *stand. jump, sit, ...*

FACIAL-EXPRESSION: *happy, excited, ...*

THOUGHT-SPEECH-BUBBLE: *think about home*

WEAR: *dressed in old pair of jeans*

TWO-PERSON-INTERACTION: *John hugged Mary*

Low-Level Spatial and Graphical Primitives: This is a grab-bag of lower level primitives such spatial relations and surface properties (colors, textures, reflectivity). These are often used to specify relations and properties on objects in the vignette (often to represent the resulting state of an object from an action) rather than to directly encode the main action of the input text.

PART-OF: one entity is part of another

POSITION: *A flower in a vase.* Note that spatial relations rely on the spatial regions and affordances on the objects. (Coyne et al., 2010) Therefore the exact realization of spatial relations, like other graphical primitives, will depend on the exact set of arguments and the affordances they provide.

POSITION-BETWEEN: a figure is between a ground1 and a ground2

ORIENTATION: an entity is facing a target object or direction

SURFACE-ATTRIBUTE: the shininess, color, texture, or transparency of an entity

LIGHT-PROPERTIES: a light source color or brightness

ENTITY-SIZE-SHAPE: the size or shape of an entity

ENTITY-STATE: an entity is folded, crumpled, shattered, etc.

We note the following: 1) These graphically primitive frames themselves can sometimes be conceptually decomposed into even finer-grained graphical operations. For example, to put a character in a given pose, it is necessary to individually orient the limbs in a certain way. So it could be argued that the set of graphical relations described above are not truly primitive. Our focus, however, is to capture the cognitively salient graphical features of a scene. We therefore handle these very low-level details in our 3D graphics subsystem rather than attempting to represent them directly with vignettes. 2) If needed, a graphical frame can specify not just an entity as an argument, but also what affordance on that entity is used. For example in USE-FIXTURE/MACHINE, the TOUCH-CONTROL affordance (such as a button or switch to turn on a piece of electronics) would typically be used. If a non-default value is needed, or there is no default, it can be specified as an argument to the graphical primitive itself. 3) The graphical primitives can also include arguments for necessary supports or containers. For example, the various GRASP primitives specify not only a GRASPED-THEME argument, but also allow a CONTAINER-FOR-THEME. This allows a single primitive to handle actions like *John held the coffee* which actually involves holding a coffee mug rather than the coffee itself. We do this for convenience. In all cases, the relations between containers and supports could be specified with separate graphical primitives.

2.2 Affordances and Spatial Tags

In order to apply graphical relations to actual 3D objects we need knowledge of the structure of those objects. For example, opening a door involves grasping the doorknob, and putting a flower in a vase involves putting the stem of the flower into the container area of the vase. This knowledge of objects is captured by the notion of *affordances*. These affordances constrain how objects and human characters interact with one another in the world. Note that fairly complex poses like riding a bicycle can be accomplished by using the FOOTHOLD (pedal), GRIP-CONTROL (handlebars), and SEAT-STRADDLING (seat) affordances. We identified the following set of affordances by examining over 2000 3D objects in our library and identifying what parts of those objects would function as affordances. The interface we used for assigning these affordances to the 3D objects is described in Section 4.3.

Human Location: WALKING-AREA, PATH, WALKTHROUGH-OPENING, DOOR-GATE, VIEWING-WINDOW, VIEWING-OPENING-AFFORDANCE

Hotspot: INSTRUMENT-HOTSPOT, MACHINE-PATIENT-AREA, SOUND-SOURCE, LIGHT-SOURCE
Container and Surface: WORK-SURFACE, MACHINE-PATIENT-AREA, STORAGE-AREA, CONTAINER-BASIN, CAP-COVER, RECEPTACLE
Self-Support: SEAT-WITH-BACK, SEAT, SEAT-STRADDLING, HANDHOLD, FOOTHOLD, HANDHOLD-FOOTHOLD, LYING-SUPPORT, ARM-REST, LEG-REST, HEAD-REST
Touch-Grip: INSTRUMENT-GRIP, CARRYING-GRIP, OPEN-CLOSE-GRIP, PUSH-PULL-GRIP, TOUCH-CONTROL, GRIP-CONTROL, PEDAL, EYEPIECE, EARPIECE, NOSEPIECE, MOUTHPIECE, INSERTION-PIECE
Functional area: OUTPUT-AREA, INPUT-AREA, DISPLAY, WRITING-AREA
Spatial regions: BASE, STEM, CANOPY, TOP-SURFACE, BOTTOM-SURFACE, WALL

3 Preparation for Action Vignette Annotation

In this section we discuss how we prepared a set of approximately 1000 core verbs and their arguments to serve as input for the action vignette annotation process. These verbs were manually chosen, using subjective judgements, to include verbs that are commonly used as well as those that are concrete in nature and hence could be readily depicted. We also specified relative priorities to further guide our annotation efforts. Some of these verbs are shown in Figure 6. The list of actual verb phrases to annotate was obtained from the British National Corpus (BNC), which we parsed with the MICA parser (Bangalore et al., 2009). An AWK script was written to extract verbs from the output along with particles, direct and indirect objects, and prepositional phrases which convey salient information about the action. These verb-argument tuples provided the skeletal sentences and valence patterns representing the typical ways these verbs would be used. It is these verb-argument tuples that would then be annotated.

3.1 Parsing

MICA (Bangalore et al., 2009) is a dependency parser that uses tree insertion grammars and supertagging. A supertag is an elementary tree of a tree grammar, associated with a lexical item. The tree insertion grammar MICA is trained on was automatically extracted from a TIG-converted version of the Penn Treebank. The parser first assigns *n*-best supertags to an input token sequence, and then extracts from the lattice of super-tags the most likely complete tree insertion grammar derivation. Each supertag carries information about the syntactic context the lexical item may occur in and other lexical information. The Mica post-processor can use this information to augment the resulting dependency parse with deep linguistic features, such as uncategorization information and voice. Verbal arguments associated with each supertag can be identified as *deep syntactic* roles (subject, object, indirect object in normalized active-voice word order). The referents of empty arguments in control and raising constructions can be identified and re-attached to their verbs. This makes extraction of verb/core-argument seeds much easier and more reliable.

3.2 Extraction and Sorting

The script works in the following way: any verb that is not marked as passive voice is recorded by the script in its lemma form, which then looks for verb particles, objects, and prepositional phrases. Verb particles, irrespective of their position in the original sentence, are placed with the verb, and the verb-particle combination is treated as a distinct lexical item. The script then searches for the direct and indirect objects of the verb. Object pronouns which refer to people are normalized to *someone*, and those referring to objects to *something*; the exception is *them*, which is ambiguous in this respect. Reflexive pronouns are similarly normalized to

oneself. Articles are normalized to *a* or *an*, and possessive adjectives to *one's*. No other noun modifiers are preserved. The script then looks for prepositions which are children of the verb, and finds their objects, making the same modifications to these objects as to direct and indirect objects. The preposition and its objects are returned as a single *prepositional phrase* unit. The direct object, indirect object, and prepositional phrase are returned after the verb in the order in which they appear in the sentence. Finally, if the script detects an infinitive verb which is subject-controlled by the verb being processed, the phrase *to do something* is added to the very end of the phrase.

There were, at times, problems with the parse which impeded the function of the extractor, prompting the addition of code in the extractor work around these errors. For example, the contraction *I'm* was parsed as two lexical items by MICA: *I* and *'m*. However, *'m* was not recognized as a form of the word *am*—itself a form of the verb *to be*—but rather was analyzed as a distinct verb. Thus, the extractor was modified to detect and fix this particular case.

After a list of verb phrases was produced from the BNC, it was pared down to a more manageable size. First, the verb phrases were sorted by frequency, so that the most common phrases appeared at the top of the list. From this, we filtered out verb phrases based on whether they were part of our list of core verbs. Some examples of final extracted verb-argument tuples:

(131 (:VERB "eat") (:DIRECT-OBJECT "something")) – 131 verb phrases with *eat* and a pronoun.
(24 (:VERB "eat") (:DIRECT-OBJECT "a meal")) – 24 verb phrases with *eat* and *meal*.

4 Annotation Methods and Tools

In this section we describe some of the different methodologies and user interfaces we have developed for defining vignettes and assigning affordances and normalized part-names to 3D objects.

4.1 Using AMT to build location vignettes

We have been investigating the use of Amazon Mechanical Turk (AMT) for building locations vignettes. The inputs to our AMT tasks are 'typical' photos of different rooms, that show large objects typical of that particular room. We carefully selected the picture from the results of image searches using Google and Bing. Turkers of each task had to be in the US and had previous approval rating of at least 99%. Restricting the location of the Turkers increases the chance that they are native speakers of English, or at least have good command of the language.

Phase 1: Collecting the functionally and visually important objects of rooms

The functionally important objects for a room are those that are required in order for the room to be recognized or to function properly. The visually important objects are those that help define the basic structural makeup of that particular room instance, such as large objects and those that are fixed in location. Examples of those objects in a kitchen can be "stove", "oven", "sink", "cabinets", and so on. After collecting the objects from several AMT tasks, we post-process them with the following steps (Rouhizadeh et al., 2011a):

1. Manual checking of spelling and converting plural nouns to singular.
2. Removing conjunctions such as "and", "or", and "/".
3. Substituting the objects belonging to the same WordNet synset with the most frequent word of the synset. ("tub", "bath", and "bathtub" ⇒ "bathtub")

4. Substituting words with major substrings in common (“night stand”, “night-stand” ⇒ “nightstand”).
5. Selecting the head nouns of compounds (“computer monitor” ⇒ “monitor”).

Phase 2: Collecting the visual properties of the rooms

Turkers should determine the room layout (diagonal or horizontal), room size (small, medium, or large), ceiling height, wall texture (painted color, wallpaper pattern, fabric, wood paneling, tile, concrete, or stone), and floor texture (tile, wood, carpeted, stone, or concrete).

Phase 3: Collecting the spatial relations between the objects

For each object **O** that is collected in phase 1 Turkers should answer the following questions: (see Figure 2)

1. Is **O** located against a wall? If so, determine the wall.
2. Is **O** near another object? If so, determine the object, determine the direction (front, back, or side), and the distance (1 ft, 2 ft, 3 ft, or 4 ft or more).
3. Is **O** supported by (i.e. on, part of, or attached-to) another object? If so, determine the object.
4. Is **O** facing another object? (e.g. chair facing a table) If so, determine the object.

We have completed phases 1 and 2 for 85 rooms and are now performing phase 3 for those rooms. To evaluate the results of phase 1, we compare the objects we collect to a gold-standard set of objects that are found in five rooms compiled by an expert. 91% of our collected objects were correct (precision) and we could gather 88% of the objects that we expected (recall).


4.2 Defining Location Vignettes using a Text-to-Scene System

We are also using WordsEye itself to define location vignettes for rooms (see Figures 3, 4). Annotators use WordsEye to textually describe rooms, and in the process see what those rooms look like. Those textual descriptions correspond to the graphical relations needed to define vignettes. We have currently defined about 50 fairly detailed rooms of different types using this method. The main advantage over the menu driven approach described in 4.1 is that the annotator can much more quickly and easily describe simple spatial relations (e.g. *the bed is against the wall and 3 feet to the right of the dresser*) than finding and filling in parameters for the objects, the relations, and their arguments on a complex set of menus. In addition, this method also has advantages over “working blind” with either menus or the purely textual descriptions described in Rouhizadeh et al. (2011c). The visual feedback of seeing the location of the room as it is described lets the annotator gauge how their specifications will actually be interpreted and depicted. In addition, the annotator can work incrementally and base additional input on an actual rendered scene rather than relying on a fleeting mental image or an interpretation of existing text or menu specifications. Furthermore, this makes it easier for annotators to use already-defined locations as a textual and visual starting point for additional variations.

Using the text-to-scene system to depict locations as they are described also serves to ensure that the inputs are well formed. All inputs are parsed and converted to a semantic representation consisting of objects and graphical primitives. As a result the input text is automatically converted into the graphical relations used by the vignette being defined. No post-processing is required. This applies not only to the relations but also to the specific object types that are used in the location. For example, if the user specifies that a chair is in a kitchen, they can pick the specific type of chair and avoid inappropriate chairs like lounge chairs and electric chairs.

Answer the following questions for each of the selected objects in this bedroom.

Bedroom (publicly available from: <http://www.republicdomain.com/doubt-bedroom>)



dresser

Is dresser located AGAINST a wall? (no)

Is dresser NEAR another object? (eg chair near a table) (no) Choose direction: (front) choose distance: (1ft)

Is dresser SUPPORTED BY (on, part of, or attached-to) another object? (floor)

Is dresser FACING another object? (eg chair facing a table) (does not have front side)

bed

Is bed located AGAINST a wall? (no)

Is bed NEAR another object? (eg chair near a table) (no) Choose direction: (front) choose distance: (1ft)

Is bed SUPPORTED BY (on, part of, or attached-to) another object? (floor)

Is bed FACING another object? (eg chair facing a table) (does not have front side)

Figure 2: Collecting spatial relations between objects in Phase 3 of the AMT task

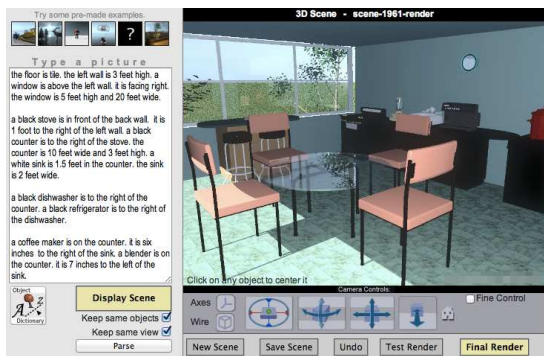


Figure 3: Kitchen location vignette defined with WordsEye



Figure 4: Other location vignettes defined with WordsEye

4.3 Assigning Part Names and Affordances

WordsEye utilizes a library of 2200 common 3D objects. The parts of these objects are often named (by the 3D artist who created them) with some abbreviated form of a normal English word. For example, the left and right tusks on one of the 3D models for an elephant are named *ltusk* and *rtusk*. In addition to normalizing part names, we need to specify which parts function as affordances. We developed a web-based user interface to assign both a normalized part name and any corresponding affordances for every part (see Figure 5).

It is important to note that we are defining parts and affordances on actual 3D objects rather than on conceptual types in the ontology. This eliminates a typical problem in ontologies where higher-level concepts define properties that may not apply to all lower level concepts. For example, in WordNet (Fellbaum, 1998), the synset *shoe* has a meronym (part designation) of *shoelace*. Not all types of shoes, however, have shoelaces. In particular, the synset *loafer* is a hyponym of *shoe*, but loafers don't have shoelaces. Since our taxonomy contains actual 3D objects, we can instead assign the exact set of parts, affordances, and other properties that apply to those specific objects. We can then infer by induction that most shoes have laces, but not all.

We have successfully annotated our entire 3D library in this manner, in the process renaming about 18,000 parts and assigning 2,400 affordances. Since our interface lets the annotator assign part names by typing in a word, there were cases of unknown or misspelled words. There were also cases of ambiguity, with multiple word senses for the same input part name. These were fixed in a post-process by automatically finding all problematic cases (those with either no known word sense or more than one) and manually assigning the correct sense.

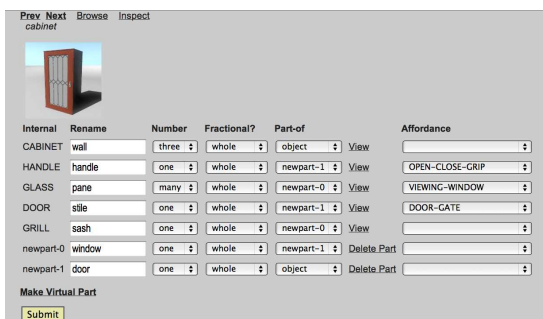


Figure 5: Parts and affordances for a door 3D object

4.4 Defining Action Vignettes

Action vignettes are specialized FrameNet frames that represent different ways of decomposing an action into graphical primitive frames. Like other frames, they can be evoked by a lexical item in its syntactic context. In order to simplify the annotation process, we are currently mapping vignettes directly to valence patterns, bypassing any corresponding FrameNet frames. We are decoupling the vignette definition process from FrameNet for a couple reasons. First, many common verbs (such as *bounce*) have no defined lexical items within FrameNet. About

17% of our core verbs have no FrameNet equivalent. Secondly, the mapping from frames to vignettes would complicate and slow down the annotation process, forcing the annotator to understand much of FrameNet in addition to our graphical primitives. These problems will be addressed in future work (see Section 5).

Annotating action vignettes involves mapping each verb-argument tuple to sets of primitive graphical frames representing how that verb (with arguments) would be depicted. See Figure 7. These decomposed tuples implicitly define new vignettes, where the tuples correspond to valence patterns that specify when that vignette is invoked. Since many tuples can be depicted in the same basic manner (albeit with different input arguments), we allow inheritance between vignettes. Inherited vignettes decompose to the same set of graphical primitives as their parent, but can override the values of the arguments. For example, *wash an apple* and *wash a pear* would invoke the same vignette, which we can think of as WASH-FRUIT-IN-SINK. The user interface supports this functionality by allowing the annotator to select and assign multiple input tuples to the same vignette.

We have so far annotated about 90 core verbs using this interface. In the process, we have specified 450 top-level vignettes and 2500 inherited vignettes (some of which override the inherited values). We have found that the original set of graphical primitives (Section 2.1) has remained fairly stable during this process, with an occasional new parameter or value type being added as new cases were encountered. We have made changes to the user interface (for example, adding different sorting keys and viewing options) based on experience using it. See Figure 8.

Action Vignette Verbs		
Priority=0		
0/0:	arrive (1478) annotated=3, inherited=35	Initial set
0/1:	ask (2462) annotated=3, inherited=25	Initial set
0/2:	believe (1165)	Initial set
0/3:	bow (99) invalid=8, annotated=5, inherited=6	9 days ago
0/4:	bring (5856) annotated=2, inherited=1	Initial set
0/5:	call (2956)	
0/6:	carry (3019) annotated=6, inherited=17, notes=1	7/26/2012
0/7:	close (1130) annotated=7, inherited=53	Initial set
0/8:	come (7679)	
0/9:	connect (891) annotated=2, notes=1	7/13/2012
0/10:	cover (2547) annotated=1, inherited=6	Initial set
0/11:	crawl (182)	
0/12:	cross (885) annotated=4, inherited=26	Initial set
0/13:	dance (309) invalid=9, annotated=6, inherited=33	
0/14:	dive (173)	
0/15:	do (6532)	
0/16:	drag (351)	
0/17:	drink (355) annotated=1, inherited=39	
0/18:	drive (2016) annotated=3, inherited=8, notes=1	7/20/2012
0/19:	drop (1356) annotated=10, inherited=26	7/9/2012
0/20:	eat (1161) invalid=4, annotated=6, inherited=155	7/10/2012
0/21:	end (1180)	
0/22:	enjoy (1584)	7/10/2012
0/23:	enter (1869) annotated=1, inherited=114	7/10/2012
0/24:	face (2060) annotated=1	7/10/2012
0/25:	fall (2544) annotated=5, inherited=9, notes=3	7/11/2012

Figure 6: Action Vignette Browser for selecting verbs and showing annotation status

Pattern: ((:SUBJ "person") (:VERB "wash") (:DIRECT-OBJECT "a fruit"))

Sub-relations Add

ENVIRONMENT Delete

TIME-OF-DAY ANY-TIME

VENUE kitchen

SUB-VENUE

POSITION-THEME-MANUALLY Delete

AGENT SUBJ (person)

THEME OBJ (a,fruit)

THEME-CONTAINER

TARGET

TARGET-LOCATION-PART CONTAINER-BAS

Semiotic status literal

Save Sub-Relations Clear all Update children

Pattern: ((:SUBJ "person") (:VERB "chop") (:DIRECT-OBJECT "a carrot"))

Sub-relations Add

ENVIRONMENT Delete

TIME-OF-DAY ANY-TIME

VENUE kitchen

SUB-VENUE

APPLY-INSTRUMENT-USING-WORKSURFACE Delete

AGENT SUBJ (person)

INSTRUMENT knife

PATIENT OBJ (a,carrot)

PATIENT-PART

PATIENT-CONTAINER chopping board

WORK-SURFACE counter

ARM-MOTION DEFAULT

Semiotic status literal

Save Sub-Relations Clear all Update children

Figure 7: Action Vignette Editor for wash fruit and chop carrot

verb=eat(1161) [annotated by alex] Browse Guidelines Load patches

Selected items: Inherit vignette Clear inheritance Mark invalid Mark valid

all invalid valid (un)labeled notes same-pattern sorted pp-sorted pp-only/sorted

12 Next Last

(13) (VERB "eat") (DIRECT-OBJECT "something") Edit

(6) (VERB "eat") (DIRECT-OBJECT "food") Add Agent

(3) (VERB "eat") (DIRECT-OBJECT "meat") Edit

(8) (VERB "eat") (DIRECT-OBJECT "meat") Edit

(24) (VERB "eat") (DIRECT-OBJECT "a meat") Edit

(23) (VERB "eat") (DIRECT-OBJECT "meat") Edit

(22) (VERB "eat") (DIRECT-OBJECT "a food") Edit

(20) (VERB "eat") (DIRECT-OBJECT "food") Edit

(21) (VERB "eat") (AUX "to do something") Edit Add Venue Add Agent

(18) (VERB "eat") (DIRECT-OBJECT "a lot") Edit Add Venue Add Agent

(17) (VERB "eat") (DIRECT-OBJECT "meat") Edit Add Venue Add Agent

(16) (VERB "eat") (DIRECT-OBJECT "a sandwich") Edit

(15) (VERB "eat") (DIRECT-OBJECT "eggs") Edit

(13) (VERB "eat") (DIRECT-OBJECT "sandwich") Edit

(12) (VERB "eat") (DIRECT-OBJECT "one dinner") Edit

(11) (VERB "eat") (DIRECT-OBJECT "something") Edit Add Venue Add Agent

(11) (VERB "eat") (DIRECT-OBJECT "one sandwich") Edit

(10) (VERB "eat") (DIRECT-OBJECT "a sandwich") Edit Add Agent

(10) (VERB "eat") (DIRECT-OBJECT "one food") Edit

(10) (VERB "eat") (DIRECT-OBJECT "one lunch") Edit

(9) (VERB "eat") (DIRECT-OBJECT "meat") Edit Add Venue Add Agent

(9) (VERB "eat") (DIRECT-OBJECT "spices") Edit Add Venue Add Agent

(8) (VERB "eat") (DIRECT-OBJECT "nothing") Edit Add Venue Add Agent

(8) (VERB "eat") (DIRECT-OBJECT "something") Edit Add Venue Add Agent

(8) (VERB "eat") (DIRECT-OBJECT "things") Edit

(8) (VERB "eat") (DIRECT-OBJECT "meats") Edit Add Venue Add Agent

(8) (VERB "eat") (PP "in a way") Edit Add Venue Add Agent

(8) (VERB "eat") (DIRECT-OBJECT "a grass") Edit Add Venue Add Agent

(8) (VERB "eat") (DIRECT-OBJECT "meat") Edit Add Venue Add Agent

(6) (VERB "eat") (DIRECT-OBJECT "an animal") Edit Add Venue Add Agent

(6) (VERB "eat") (DIRECT-OBJECT "meat") Edit Add Venue Add Agent

Pattern: ((:SUBJ "person") (:VERB "eat") (:DIRECT-OBJECT "food"))

Vignette Arg Pattern Val Assigned Val

ARG-0 (TIME-OF-DAY) --- ANY-TIME

ARG-1 (VENUE) --- kitchen,dining room,cafeteria,resta

ARG-2 (AGENT) --- "person"

ARG-3 (STANCE) --- SIT-ON-SEAT

ARG-4 (SUPPORT-FIXTURE) --- chair

ARG-5 (AT-FIXTURE) --- table

ARG-6 (INSTRUMENT) --- fork

ARG-7 (PATIENT) --- "food"

ARG-8 (PATIENT-CONTAINER) --- plate

Save argument assignments

Sub-relations Add

ENVIRONMENT Delete

TIME-OF-DAY ARG-0 (ANY-TIME)

VENUE ARG-1 (kitchen,dinr)

STANCE Delete

AGENT ARG-2 (person)

STANCE ARG-3 (SIT-ON-SEA)

SUPPORT-FIXTURE ARG-4 (chair)

AT-FIXTURE ARG-5 (table)

APPLY-INSTRUMENT-USING-WORKSURFACE Delete

AGENT ARG-2 (person)

INSTRUMENT ARG-6 (fork)

PATIENT ARG-7 (food)

PATIENT-PART

PATIENT-CONTAINER ARG-8 (plate)

WORKSURFACE ARG-5 (table)

Figure 8: Action Vignette Editor. Left side shows vignette for the current input pattern. Top-left shows vignette arguments and bottom-left shows graphical decomposition applied to those arguments. Right side shows input tuples. Color coding is used on tuples to denote inheritance relations and annotation status. Different filtering and sorting options allow the annotator to focus on particular valence patterns.

5 Conclusion and Future Work

In this paper we have presented approaches to acquiring real-world knowledge to be used in a text-to-scene system. The core of our approach is embodied in the notion of vignettes. Vignettes are frames that are decomposable into grounded graphical relations. To implement vignettes we have defined a specific set of graphical primitives that allow us to map between high-level to low-level semantics. These graphical primitives enable a wide variety of scenes to be specified with a sufficient level of detail. In order to apply these graphical primitives to 3D objects we have defined a set of affordances representing the structure of those objects and how they are used and manipulated. Finally, building on this framework of vignettes, graphical primitives, and affordances, we have developed several methods for populating our resource with both locations and action vignettes. For action vignettes, this involved preparing a corpus of verb-argument tuples to be used as input data and developing tools to annotate that data with vignettes. To define location vignettes we used both AMT crowdsourcing methods and WordsEye, our text-to-scene system, as an annotation tool itself. Our resulting resource (called VigNet) is a lexically-oriented knowledge-base with lexical inputs mapping to vignettes, which in turn map to graphical primitives and actual 3D objects. VigNet will be made publicly available at <http://www.cs.columbia.edu/speech/text2scene>.

Future work includes finishing the vignette annotation process. In addition, action vignettes, as defined in the user interface, will require some amount of post-processing in order to be used. In particular, we need to handle unknown words and word sense ambiguity issues for those words that were typed in by the annotator. Those will be normalized and disambiguated in a separate semi-automatic pass as we did for part names (Section 4.3). We will also perform a separate post-processing task to link vignettes to their corresponding FrameNet frames.

One main challenge in using vignettes is that there won't be a vignette to match every possible input. Instead, it will be necessary to generalize the input arguments to find the closest vignette. For example, a vignette defined for *wash an apple* (using a sink) can be applied to washing any small round fruit. This is partially addressed by annotators listing multiple possible values to fill arguments when defining the vignette. In general, however, finding the closest vignette will involve estimating the semantic distance between the arguments of the candidate vignettes and those of the input sentence. To do this, we will leverage information in our ontology that specifies the sizes, shapes, substances, and other semantic properties of all 3D objects and their parents. We also note that vignettes, themselves, can explicitly specify arbitrarily complex sets of constraints and restrictions on their arguments to help make these matches.

A second major challenge in using vignettes is to compose actions vignettes with location vignettes in the course of text-to-scene generation. We intend to accomplish this as follows. If an action is mentioned, then the default venue for that action will evoke a set of possible location vignettes. Any affordances required by the action will be unified with those provided by the location. For example, the vignette for *chop carrots*, might supply a default VENUE of a kitchen and APPLY-INSTRUMENT-USING-WORKSURFACE. The kitchen vignette includes a counter and kitchen table, both of which provide a WORK-SURFACE affordance.

Acknowledgments

This work was supported in part by NSF IIS- 0904361. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

- Bangalore, S., Boullier, P., Nasr, A., Rambow, O., and Sagot, B. (2009). Mica: a probabilistic dependency parser based on tree insertion grammars application note. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 185–188. Association for Computational Linguistics.
- Coyne, B., Bauer, D., and Rambow, O. (2011a). VigNet: Grounding Language in Graphics using Frame Semantics. In *Workshop on Relational Models of Semantics (RELMS) at ACL*.
- Coyne, B., Schudel, C., Bitz, M., and Hirschberg, J. (2011b). Evaluating a Text-to-Scene Generation System as an Aid to Literacy. In *Workshop on Speech and Language Technology in Education (SlaTE) at Interspeech 2011*.
- Coyne, B. and Sproat, R. (2001). Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496, Los Angeles, CA, USA.
- Coyne, B., Sproat, R., and Hirschberg, J. (2010). Spatial relations in text-to-scene conversion. In *Computational Models of Spatial Language Interpretation, Workshop at Spatial Cognition 2010*, pages 9–16, Mt. Hood, OR, USA.
- Fellbaum, C. (1998). *WordNet: an electronic lexical database*. MIT Press.
- Gibson, J. (1977). The Theory of Affordances. In Shaw, R. and Bransford, J., editor, *Perceiving, acting, and knowing: Toward an ecological psychology*, pages 67–82. Erlbaum, Hillsdale, NJ.
- Haugeland, J. (1995). Mind embodied and embedded. In HounG, Y. and Ho, J., editors, *Mind and Cognition*. Academia Sinica, Taipei.
- Jackendoff, R. (1985). *Semantics and cognition*, volume 8. The MIT Press.
- Kipper, K., Dang, H. T., and Palmer, M. (2000). Class-Based Construction of a Verb Lexicon. In *Proceedings of AAAI 2000*, Austin, TX.
- Miller, G. and Johnson-Laird, P. (1976). *Language and perception*. Belknap Press.
- Norman, D. A. (1988). *The Psychology of Everyday Things*. Basic Books.
- Rosch, E. (1973). Natural categories. *Cognitive psychology*, 4(3):328–350.
- Rouhizadeh, M., Bauer, D., Coyne, B., Rambow, O., and Sproat, R. (2011a). Collecting spatial information for locations in a text-to-scene conversion system. In *Proceedings of the Workshop on Computational Models of Spatial Language Interpretation and Generation (CoSLI 2011)*, Boston, MA.
- Rouhizadeh, M., Bowler, M., Sproat, R., and Coyne, B. (2010). Data collection and normalization for building the scenario-based lexical knowledge resource of a text-to-scene conversion system. In *Proceedings of SMAP 2010: 5th International Workshop on Semantic Media Adaptation and Personalization*, pages 25 –30, Limassol, Cyprus.

Rouhizadeh, M., Bowler, M., Sproat, R., and Coyne, B. (2011b). Collecting semantic data by Mechanical Turk for the lexical knowledge resource of a text-to-picture generating system. In Bos, J. and Pulman, S., editors, *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, The University of Oxford.

Rouhizadeh, M., Coyne, B., and Sproat, R. (2011c). Collecting semantic information for locations in the scenario-based lexical knowledge resource of a text-to-scene conversion system. In König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R., and Jain, L., editors, *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6884 of *Lecture Notes in Computer Science*, pages 378–387. Springer Berlin / Heidelberg.

Ruppenhofer, J., Ellsworth, M., Petruck, M., Johnson, C. R., and Scheffczyk, J. (2010). *Framenet II: Extended Theory and Practice*. ICSI Berkeley.

Schank, R. C. and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding*. Earlbaum, Hillsdale, NJ.

Sproat, R. (2001). Inferring the environment in a text-to-scene conversion system. In *Proceedings of The First International Conference on Knowledge Capture*, pages 147–154, Victoria, BC, Canada.