3D Scene Retrieval from Text

Gireesh Singh Thakurathi[1], Melvin Thomas [2], and Haresh Savlani[3]

[1]Gireesh Singh Thakurathi, Computer Engineering/ Thadomal Shahani Engineering College/ Mumbai University, Ghatkopar West, Mumbai - 400084, Maharashtra, India
[1]gireeshsinghthakurathi@gmail.com
[2]Melvin Thomas, Computer Engineering/ Thadomal Shahani Engineering College/ Mumbai University, Mulund West, Mumbai- 400080, Maharashtra, India
[2]melvinthomas31@gmail.com
[3]Haresh Savlani, Computer Engineering/ Thadomal Shahani Engineering College/ Mumbai University, Borivali West, Mumbai – 400092, Maharashtra, India
[3]hareshsavlani@gmail.com

ABSTRACT

Translating elusive ideas or statements into visual scenes is quite a daunting task. First the thought has to be laid out explicitly and clearly in the form of a written statement which acts as the foundation for the conversion. Then a professional strives to create a mental image of the given statement. Finally, another professional starts placing models according to the abstract image created in the previous stage in a model rendering software. Thus, converting a single text to corresponding visual element is a rather challenging task. This paper will focus on automating this entire transformation. It promises to convert any arbitrary descriptive text into a representative scene. The proposed system parses a user written input text, extracts information using Natural Language Processing (NLP) and tags relevant units. It then associates every object with model and places them according to the derived relations and spatial dependencies. Ultimately the user can make changes and minor adjustments to the final scene using Blender[1] in-build controls.

Index Term— Text, scene, NLP, dependency, parser, supporter, dependent, preposition, bounding box, 3d models, refinement, POS tags, heuristics, collision.

1. INTRODUCTION

Natural language is an easy and effective medium for describing visual ideas and mental images. It is a tool that allows people to describe visual scenes in a straightforward manner. Automatic generation of scene by using text descriptions as input offers an efficient approach to human computer interaction with graphics and could speed up the whole generating process. It also makes graphics more accessible to users in non-graphics domains. Thus, foreseeing the emergence of language-based scene generation systems to let ordinary users quickly create scenes without having to learn special software, acquire artistic skills, or even touch a desktop window-oriented interface.

The paper explains the existing solutions, implementation of the proposed system in detail, its limitations and future work.

2. Existing Solution

Already quite a few projects investigated the field of natural language input for creating virtual environments.

2.1. SHRDLU program

The SHRDLU program was one of the earliest systems that were 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. [2]

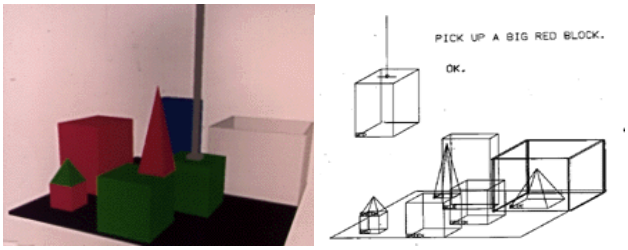


Fig-1: SHRDLU program

2.2. Wordseye

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. [3] 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. [4]

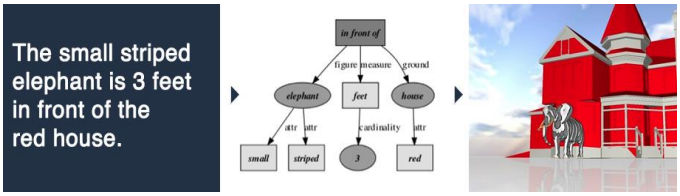


Fig-2: Wordseye

2.3. Text to Scene Generation

The Text to Scene Generation project aims to explore how to automatically generate 3D scenes from a natural text description. [5] When describing a scene, people will often omit important common-sense knowledge about the placement of objects. For instance, it is uncommon for people to state that chairs are usually on the floor and upright, and that you eat a cake from a plate on a table. This project attempts to learn such knowledge from a dataset. [6]

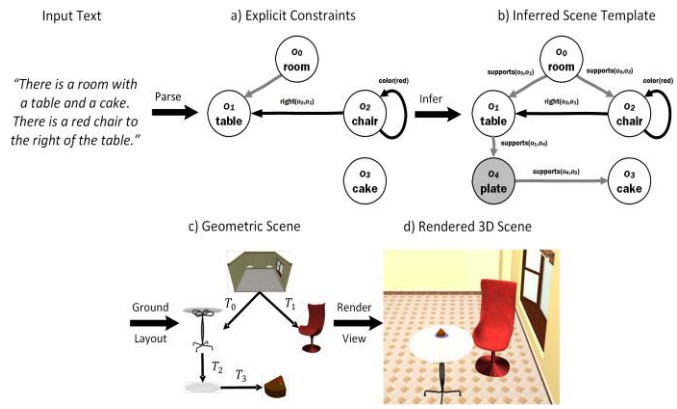


Fig-3: Text to scene generation [7]

Most previous work paid more attention to the language engine, but less attention to the graphics engine which was just designed to present scene with poor visual effects.

3. Implementation Details

The technical execution of the system has been divided into 3 main phases namely,

Phase 1: Information extraction and refinement

Phase 2: Creation of directed graph

Phase 3: Rendering virtual environment

Each phase has been further subdivided into further small steps as follows.

3.3. Phase I

This phase consists of extracting information which will be needed for the smooth execution of later stages. Extracted information belonging to the same data type are later saved in similar data structures for future reference.

3.1.1. Part of Speech Tagging

The input sentence is then divided into tokens i.e., the sentence is divided into atomic units(words) and all the words as tagged and classified according to their type (nouns, pronouns, determiner, adverbs, etc.). The technology used for tokenizing the input descriptive text is Stanford Tagger from Stanford CoreNLP package. [8]".

3.1.2. Extracting Dependencies

A dependency parser analyzes the grammatical structure of a sentence, establishing relationships between "head" words and words which modify those heads. The figure below shows a dependency parse of a short sentence. The arrow from the word moving to the word faster indicates that faster modifies moving, and the label advmod assigned to the arrow describes the exact nature of the dependency.

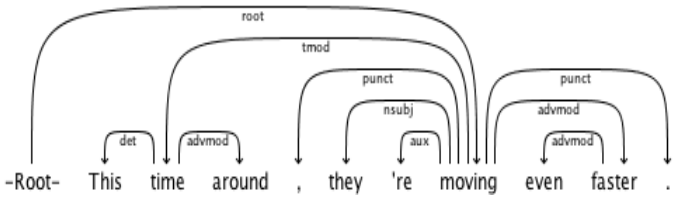


Fig-4: Dependency parser

The parser provides Universal Dependencies as well as phrase structure trees. Typed dependencies are otherwise known grammatical relations.

3.1.3. Spatial Prepositions

Prepositions that indicate a spatial relationship explain where one object is in relation to another. Prepositions are grouped according to their meaning.

PREPOSITIONAL PHRASES/ PREPOSITIONS	GROUPED INTO
Beneath, below, under, down, underneath	Under
Up ,over,aboard	Above
Near, close to, next to, about, against, beside, round, around, by, at, aside	Near
Away	Away
In, inside, into, within, in between, between, through, amongst, among, amidst, amid, alongside, along, with, together	Inside
Outside, beyond, out	Outside
Toward, ahead, across, astride, front	Ahead
Apart, opposite, behind	Behind

Table-1: Grouped Prepositions

3.1.4. Refinement

This step consists of fetching the core elements needed by the system for creation of a 3D scene. Supporters, dependents and prepositions are extracted from the sentences using the dependencies and POS tags which are then saved in a text file for creation of directed graph.

3.2. Phase II

This directed graph is used for determining the relationships between objects. In this system, the nodes represent various objects and the links between them represent their dependency. There are 6 rules for creation of directed graph.

Algorithm for creation of directed graph

INPUT: Supporters, Dependents, Dependencies, Prepositions.

OUTPUT: Directed Acyclic Graph.

3.2.1. Rule 1

New object is added

If (object node does not exist)

Create new dependent or supporter node and link it using the preposition.

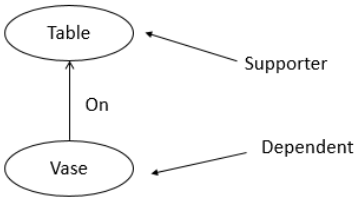


Fig-5: Rule 1

3.2.2. Rule 2

A supporter may serve for various dependents and therefore

If (supporter is not present)

Then add new object node and link it using preposition.

or else

Abort to avoid repetition.

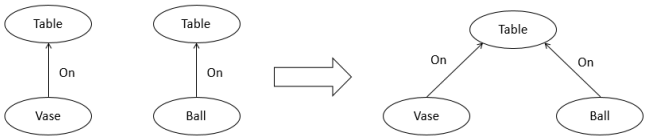


Fig-6: Rule 2

3.2.3. Rule 3

Dependents may depend on different supporters and therefore

If (dependent is not present)

Then add new object node and link it using preposition.

or else

Abort to avoid repetition.

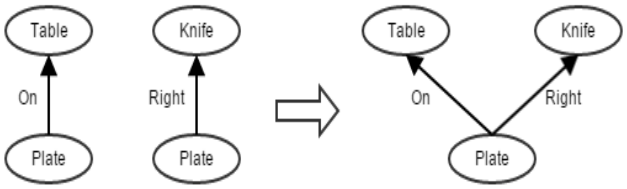


Fig-6: Rule 3

3.2.4. Rule 4

Supporter POSI qualifies to be a dependent POSI

If (supporter POSI acts as a dependent POSI)

Link supporter POSI now acting as dependent to its supporters.

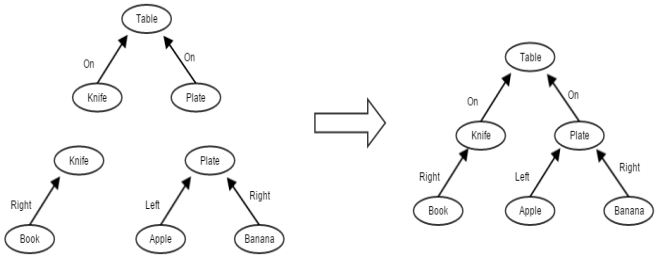


Fig-7: Rule 4

3.3.5. Rule 5

Dependent POSI qualifies to be a supporter POSI

If (dependent POSI acts as a supporter POSI)

Link dependent POSI now acting as supporter to its dependents.

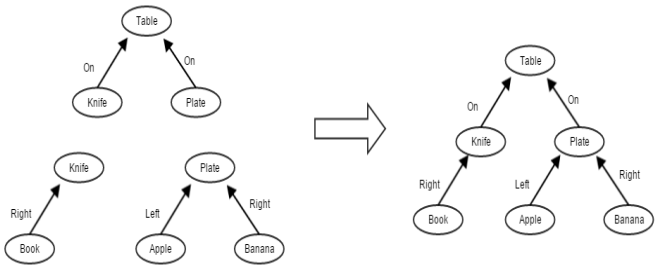


Fig-8: Rule 5

3.2.6. Rule 6

A cycle is detected in the directed graph

If (a cycle is formed)

Create another dependent node as the new child of the parent node.

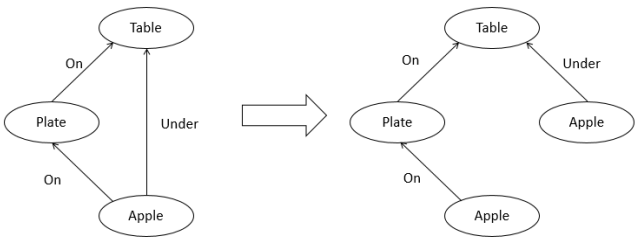


Fig-9: Rule 6

3.3. Phase III

This phase concentrates on importing the object models and placing them onto the rendering platform like Blender in a way which is dictated by the information extracted from the initial phases. Wordnet is used to pick the correct model from the database and bounding box coordinates are used to avoid collisions between neighboring objects.

3.3.1. Algorithm for importing and placing object according to spatial preposition

Input: 3D models.

Output: 3D scene as per input descriptive text.

If (spatial preposition == “on”)

Place new model just above the max height of the previous model.

If (spatial preposition == “under”)

Place new model below the base of the previous model.

If (spatial preposition == “above”)

Place above max height with randomly generated offset.

If (spatial preposition == “near”)

Place new model in any one of the four randomly chosen directions.

If (spatial prepositions == “away”)

Same as near but with a comparatively large offset.

If (spatial preposition == “inside”)

Place new model inside the previous (hollow) model.

If (spatial preposition == “outside”)

Place the new model just outside the previous model.

If (spatial preposition == “ahead”)

Place the new model to the north of the previous model.

If (spatial preposition == “behind”)

Place the model to the south of the previous model.

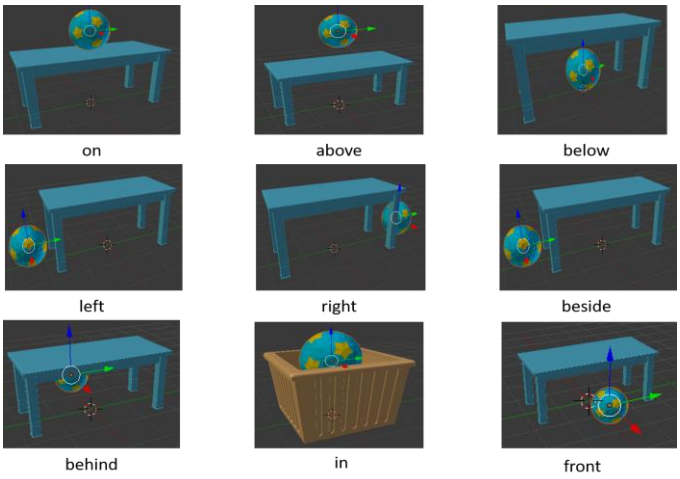


Fig-10: Spatial preposition interpretation

3.3.2. Bounding box coordinates, collision detection and resolution

It is used to determine the boundaries of the object so that it can be used to detect collisions. Once collision is detected. The current object is moved in a particular direction (using randomizer) point by point until the collision is resolved.

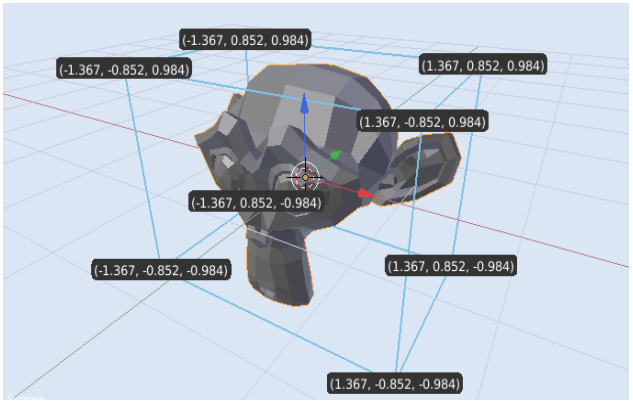
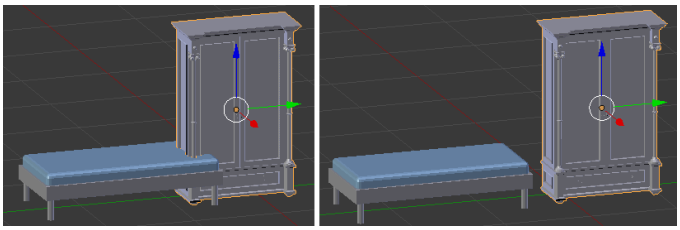


Fig-11: Bounding Box



(a) (b)

Fig-12: (a) Collision detection (b) Collision resolution

4. Limitations

A. The objects in a single sentence are in the form of supporter-dependent pairs. and linked using a preposition.

B. Each sentence has to be formed using the listed objects and preposition.

C. Each sentence should consist a maximum of one supporter and two dependents only.

D. The objects in the sentences should share some relationship between each other.

E. The system does not respond to verbs, adjectives, adverbs, colors, size or quantity.

F. Use of conjunctions is allowed.

G. System cannot recognize named entities.

H. As the complexity of the scene increases, the chances of collision rise.

5. Future Work

More can be contributed by creating a more realistic scene by incorporating heuristics making the bounding box more efficient.

5.1. Naturalness

The system would improve the appearance of a created 3D scene by applying the following two positioning heuristics.

5.1.1. Distance Heuristic

The first heuristic is used to ensure that any two objects are not placed too close or far away from each other.

5.1.2. Rotation Heuristics

This heuristic applies a little random rotation to all of the occurring objects within a scene in order to achieve an “untidy” appearance.

Consider the example:

The ball is on the table. The box is to the left of the table. The dustbin is in front of the table. The bird is above the ball. The chair is to the right of the table.

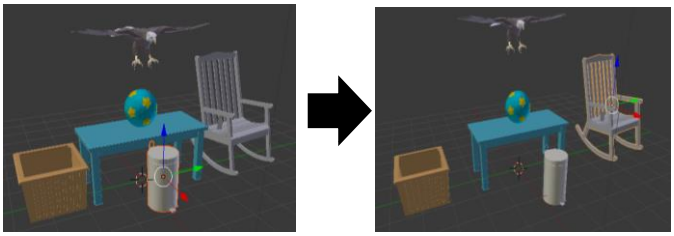


Fig-13: Imparting naturalness to scene

5.5.2. Minimum Bounding Box

Collision and wrong placements appear due to limitations of a bounding box. It can be avoided by implementing object-tight bounding boxes or switching to triangle-based collision detection. Unfortunately, this would increase computation time. [9]

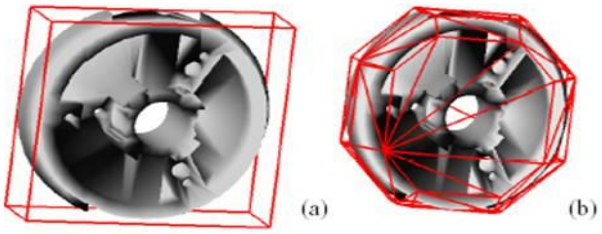


Fig-14: (a) Bounding Box (b) Minimum Bounding Box

5.5.3. Positioning

Although the positioning system is stable, misplacements of objects may still occur. 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. Using a physics engine for solving this problem thereby maintaining the physical properties of rigid body.

Consider the example:

The vase is on the table. The ball is on the table. The candle is on the table. The box is on the table.

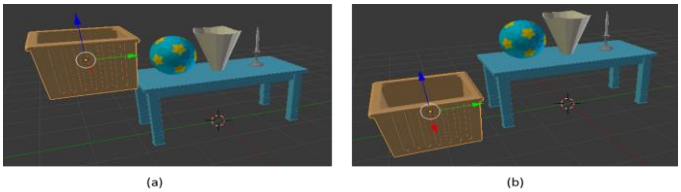
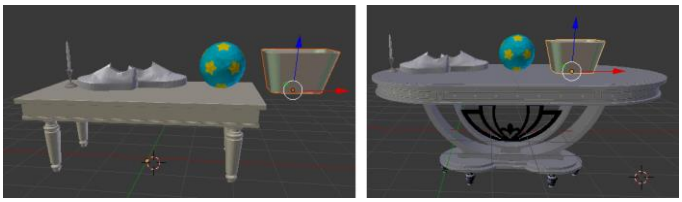


Fig-14: (a) Wrong placement (b) Misplaced objects falling on the ground due to the underlying physics engine

5.5.4: Updated database and system

In the previous case, the surface of the table fell short to accommodate all the objects without collision resolution among them and repositioning the box on the ground couldn't do perfect justice to the input text. In such a scenario, a table with a larger surface area that could accommodate all the objects would make more sense, for that the database should be quite larger as compared to the existing one and the system should be intelligent enough to import the appropriate version of the object or the objects could be scaled in real life as per the need.



(a) (b)

Fig-15: (a)Incorrect placement (b) Correct placement due to new version of table imported

6. Conclusion

I believe that the system represents a unique approach to creating scenes and images. It enables a user to quickly generate virtual environments by using natural language as input. The system will be an asset in many cases, providing interesting and surprising interpretations, and when users want to control a depiction more precisely, they can adjust their language to better specify the exact meaning and graphical constraints they envision. I believe that the low overhead of language-based scene generation systems will provide a natural and appealing way for everyday users to create 3D scenes and express themselves. In its current state, the system is only at its first step toward the goal. There are many areas where the capabilities of the system need to be improved, such as: Improvements in the coverage and robustness of the natural language processing, language input via automatic speech recognition rather than text; a larger inventory of objects, poses, atomic rules, and states of objects; mechanisms for depicting materials and textures; mechanisms for modifying geometric and surface properties of object parts; environments,

activities, and common knowledge about them; shape deformation and natural phenomena.

7. References

[1] B. Foundation, "blender.org", blender.org, 2016. [Online]. Available: https://www.blender.org/. [Accessed: 02- Aug- 2016].

[2] "SHRDLU", Hci.stanford.edu, 2016. [Online]. Available: http://hci.stanford.edu/winograd/shrdlu/. [Accessed: 2- Aug- 2016].

[3] "WordsEye", Wordseye.com, 2016. [Online]. Available: http://www.wordseye.com/. [Accessed: 25- Aug- 2016].

[4] B. Coyne and R. Sproat, "Wordseye", www.cs.columbia.edu, 2016. [Online]. Available: http://www.cs.columbia.edu/~coyne/images/wordseye_siggraph.pdf. [Accessed: 07- Sept- 2016].

[5] A. Chang, W. Monroe and M. Savva, "Text to 3D Scene Generation with Rich Lexical Grounding", nlp.stanford.edu, 2016. [Online]. Available: https://nlp.stanford.edu/pubs/chang-acl2015-lexground.pdf. [Accessed: 27- Sept- 2016].

[6] A. Chang, M. Savva and C. Manning, " Semantic Parsing for Text to 3D Scene Generation", nlp.stanford.edu, 2016. [Online]. Available: https://nlp.stanford.edu/pubs/scenegen-sp2014.pdf. [Accessed: 27- Sept- 2016].

[7] "The Stanford Natural Language Processing Group", Nlp.stanford.edu, 2016. [Online]. Available: http://Nlp.stanford.edu. [Accessed: 15- Oct- 2016].

[8] "Stanford CoreNLP – Core natural language software | Stanford CoreNLP", Stanfordnlp.github.io, 2016. [Online]. Available: https://stanfordnlp.github.io/CoreNLP/. [Accessed: 18- Oct- 2016].

[9] "Minimum bounding box algorithms", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Minimum_bounding_box_algorithms. [Accessed: 5- Sept- 2016].

