Scene Generation From Text Using Natural Language Processing

Nikhil Viradiya[1], Ambika Prasad Tripathi[2], Dhruv Nagpal[3], Yash Dixit[4], Prof. K. Jayamalini[5]
[1, 2, 3, 4, 5] Dept of Computer Engineering
[1, 2, 3, 4, 5] Shree L.R. Tiwari College of Engineering, Thane

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. Our system associates every object with an appropriate 3D model and evaluates spatial dependencies of the entities. Finally, a physics engine is used to render a realistic and interactive 3D scene which enables the user to actively manipulate the stage setup. The system relies on a database of models and poses to depict entities and actions*

Keywords- Scene Generation from Text, Natural Language Processing, Text, Raw Text, Query

I. INTRODUCTION

The Scene Generation from Text system presents a framework which automatically generates visual descriptive scenes from natural language. Nowadays graphics are used in many applications, such as cartoons, animations and games. However, creating graphics is a difficult and time-consuming task. Typical scene modelling 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.

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, we foresee 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.

II. LITERATURE REVIEW

Here we will elaborate the aspects like the literature survey of the project and what all projects are existing and been actually used in the market which the makers of this project took the inspiration from and thus decided to go ahead with the project covering with the problem statement.

The paper [1] is "Frame Semantics in Text-to-Scene Generation "by Bob Coyne, Owen Rambow, Julia Hirschberg, and Richard Sproat. In this paper they describe some of our recent work designing, building, and utilizing the SBLR in order to produce a system with much broader and robust coverage. Also they describe related work with natural language interfaces to graphics.

The paper [2] is "Creation of Scene from Raw Text" by Sneha N. Dessai, Rachel Dhanaraj. In this paper the main task in text to scene system is to extract explicit and infer implicit constraints. Clay et al [2] discusses one of such system where input is expressions in the form of Put (X, P, Y). X and Y represented objects and P was spatial preposition.

The paper [3] is "Text to Scene Generation with Rich Lexical Grounding" by Angel Chang, Will Monroe, M anolis Savva, Christopher Potts and Christopher D. Manning. In this paper The ability to map descriptions of scenes to geometric representations has a wide variety of applications; many creative industries use scenes.

III. SYSTEM ARCHITECTURE

In the text to 3D scene generation task, the input is a natural language description, and the output is a 3D representation of a plausible scene that fits the description and can be viewed and rendered from multiple perspectives. A naive approach to scene generation might use keyword search to retrieve 3D models. However, such an approach is unlikely to generalize well in that it fails to capture important object attributes and spatial relations. In order for the generated scene to accurately reflect the input description, a deep understanding of language describing environments is necessary. Our system architecture is shown in below figure.

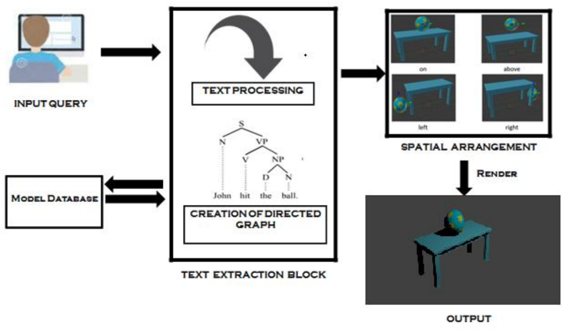


Fig- Architecture of SGFT.

IV. IMPLEMENTATION

A. System Description

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

Phase 1: Information extraction and refinement.

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.

For information extraction, Stanford CoreNLP Toolkit is used because of the following:

- An integrated toolkit with a good range of grammatical analysis tools
- Fast, reliable analysis of arbitrary texts
- The overall highest quality text analytics
- Support for a number of major (human) languages
- Interfaces available for various major modern programming languages
- Ability to run as a simple web service

Phase 2: Creation of directed graph.

A directed graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are directed from one vertex to another. A directed graph is sometimes called a digraph or a directed network. In contrast, a graph where the edges are bidirectional is called an undirected graph. When drawing a directed graph, the edges are typically drawn as arrows indicating the direction.

In our system, the nodes represent various objects and the links between them represent their dependency.

Phase 3: Rendering virtual environment.

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

B. Methodologies of SGFT

INPUT: Descriptive Text from user.
OUTPUT: 3D scene as per input descriptive text.

Step 1: Accept input descriptive text from the user.
Step 2: POS Tagging.
Step 3: Dependency Extraction.
Step 4: Recognizing supporter, dependent and preposition.
Step 5: Creation of Directed Acyclic Graph.
Step 6: Import models from database.
Step 7: Applying Bounding Box Algorithm to each model.
Step 8: Object Placement according to spatial prepositions.

C. Algorithms Used in SGFT

Algorithm 1: Directed Acyclic Graph

INPUT: Supporter, Dependents, Dependencies, Prepositions.
OUTPUT: Directed Acyclic Graph.

```
If (object node does not exist)
        Create new dependent or supporter node and link it using the preposition.
```

```
If (supporter is not present)
        Then add new object node and link it using preposition.
or else
        Abort to avoid repetition.

If (dependent is not present)
        Then add new object node and link it using preposition.
or else
        Abort to avoid repetition.

If (supporter POSI acts as a dependent POSI)
        Link supporter POSI now acting as dependent to its supporters.

If (dependent POSI acts as a supporter POSI)
        Link dependent POSI now acting as supporter to its dependents.

If (a cycle is formed)
        Create another dependent node as the new child of the parent node.
```

Algorithm 2: Object Placement 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 new model above the max height of the previous model with some dynamically generated offset depending upon the sizes of the two models.

If (spatial preposition == "near")

        Place new model in any one of the four randomly chosen primary faces (north, south, east, west) of the previous object with some dynamically generated small offset depending upon the sizes of the two models.

If (spatial prepositions == "away")

        Place new model in any one of the four randomly chosen primary faces (north, south, east, west) of the previous object with some dynamically generated comparatively large offset depending upon the sizes of the two models.

If (spatial preposition == "inside")

        Place new model inside the previous model (previous model must have a hollow core).

If (spatial preposition == "outside")

        Place the new model just outside the previous model in any one of the four randomly chosen primary faces (north, south, east, west) of the previous object.

If (spatial preposition == "ahead")

        Place the new model to the north of the previous model (+y direction) with some dynamically generated offset depending upon the sizes of the two models.

If (spatial preposition == "behind")

        Place the model to the south of the previous model (-y direction) with some dynamically generated offset depending upon the sizes of the two models.
```

V. RESULTS

The following are the screenshots of the User interface and Output: -

Consider an example:

The ball is on the table.

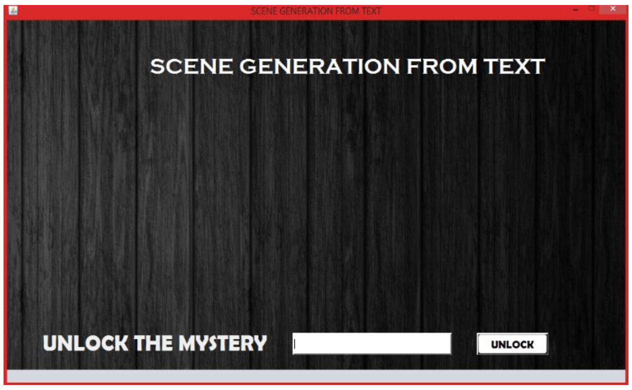


Fig (1): Unlock system of SGFT

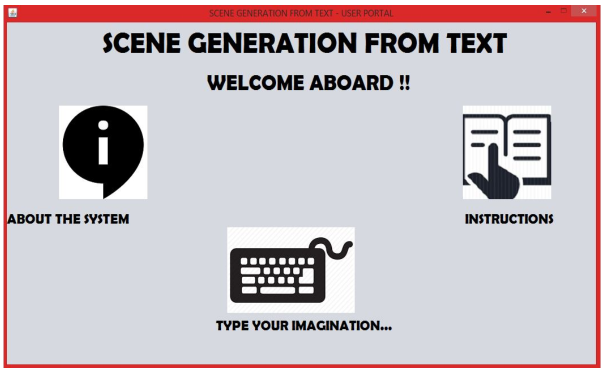
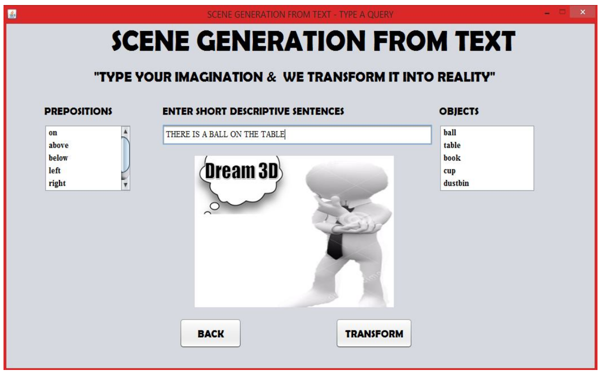


Fig (2): Option Panel OF SGFT


SCENE GENERATION FROM TEXT - TYPE A QUERY
SCENE GENERATION FROM TEXT
"TYPE YOUR IMAGINATION & WE TRANSFORM IT INTO REALITY"
Dream 3D
BACK
TRANSFORM

PREPOSITIONS: on, above, below, left, right

ENTER SHORT DESCRIPTIVE SENTENCES: THERE IS A BALL ON THE TABLE

OBJECTS: ball, table, book, cup, dustbin

Fig (3): Input Query

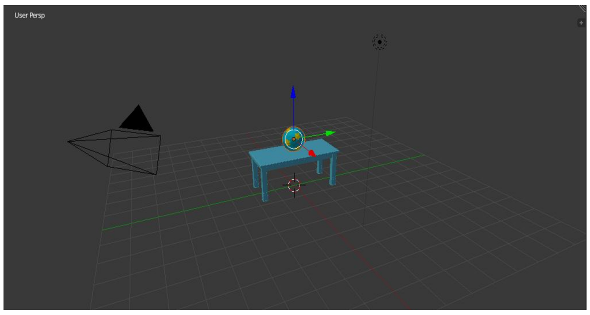


Fig (4): Generated Scene

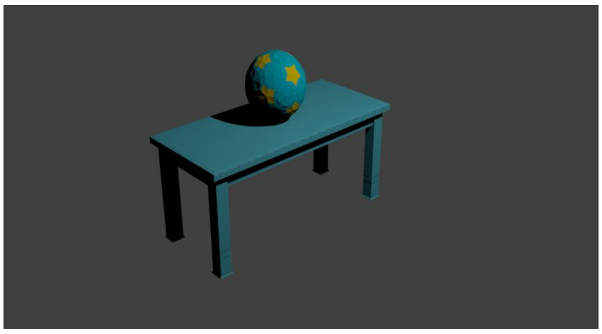


Fig (5): Rendered Scene

VI. CONCLUSION

We believe our system represents a unique approach to creating scenes and images. Our 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 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.

REFERENCES

[1] http://www.ling.upenn.edu/courses/Fall/ling/penn_treebank_pos.html
[2] https://nlp.stanford.edu/projects/text2scene.shtml
[3] https://www.wordseye.com
[4] http://nlp.stanford.edu:8080/corenlp/ process
[5] https://stackoverflow.com/questions/771918/how-do-i-do-word-stemming-or- lemmatization
[6] https://www.youtube.com/watch?v=ALfl4tebiQM
[7] http://www.cs.columbia.edu/~coyne/related-research.html
[8] http://www.cs.columbia.edu/~coyne/related-research.html