From a Textual Narrative to a Visual Story

Imran Shafiq Ahmad\(^{(E3)}\), Havish Kadiyala\(^{(E3)}\), and Boubakeur Boufama\(^{(E3)}\)

School of Computer Science, University of Windsor, Windsor, ON N9B 3P4, Canada
\{imran,kadiyalh,boufama\}@uwindsor.ca

Abstract. Much of our daily learning is done through visual information. Visual information is an indispensable part of our life and tends to convey a lot more details than either speech or text. A visual portrayal of a story is generally more appealing and convincing. It is also useful in a variety of applications, such as an accident/crime scene analysis, education and treatment of various psychological or mental disorders like Post-Traumatic Stress Disorder (PTSD). Some individuals develop PTSD due to their exposure to some dangerous or shocking life experience, such as military conflict, physical or sexual assault, traffic or fire accident, natural disasters, etc. People suffering from PTSD can be treated using Virtual Reality Exposure Therapy (VRET), where they are immersed in a virtual environment to face feared situations that may not be safe to encounter in real life. In addition, generated 3D scenes can also be used as a visual aid for teaching children. Since creating 3D context and scenarios for such situations is tedious, time-consuming and requires special expertise in 3D application development environments and software, there is a need for automatic 3D scene generation systems from simple text descriptions. In this paper, we present a new framework for creating 3D scenes from a user-provided simple text. This proposed framework allows us to incorporate motion as well as special effects into the created scenes. In particular, the framework extracts the objects and entities that are present in a given textual narrative as well as spatial relationships. Depending on the description, it then creates either a 3D scene or a 3D scene with corresponding animation. This framework allows creation of a visualization using a set of pre-existing objects using Autodesk\(\text{Maya}\)\(^{\text{R}}\) as an implementation environment.

Keywords: 3D scene generation from text · Spatial relations · Text to scene creation · Multimedia · Computer graphics · Animation

1 Introduction

A connected series of events, either real or fictional, is generally presented as a story using textual description either in written or oral form. Visual story telling, on the other hand, is a presentation of the same information using simple visual
aids and has been around for years. The old English proverb says “A picture is worth a thousand words” because a picture can concisely describe a concept, a situation, or an abstract idea which may need many words to get the point across. Many different application areas benefit from visual representation of a situation that is generally described in words. Examples of such areas include, but are not limited to, creation of an accident/crime scene for detailed analysis and investigation, education, training, proof-of-concept marketing or advertising campaigns, and treatment of many complex psychological or mental disorders, such as Post Traumatic Stress Disorder (PTSD).

Post-traumatic stress disorder (PTSD) is a serious mental disorder. In many cases, it is a result of either experiencing or witnessing some terrifying event. Symptoms may include a constant feeling of fear, flashbacks, nightmares and severe anxiety, as well as uncontrollable thoughts about the event [1,14]. Virtual Reality Exposure Therapy (VRET) is a possible therapeutic treatment for patients suffering from PTSD [1] and is said to help the emotional plight of patients during exposures. Since it is neither possible to recreate life-threatening incidents nor to know what exactly a person with PTSD may have felt during the incident, patients suffering from PTSD can be immersed in a virtual environment and the health practitioners can observe their emotional and mental state during any such incident. VRET has been used in treating some war veterans experiencing PTSD [13]. Creating a virtual environment requires a lot of time because scenarios faced by individuals often vary significantly. For this reason, it necessary to automatically create 3D scenes from simple text descriptions. Similarly, many crime scene and accident investigations require 3D creation of an actual scene as narrated by witnesses for detailed analysis and for helping understand complex situations during legal proceedings. 3D scenes or 3D animations of such instances may not only provide a very clear picture, but can also be used to help them understand how the incident may have unfolded.

To describe a situation or to narrate a story, the medium of communication is generally some natural language. The individual(s) reading or listening to this story generally create(s) a visual image of the objects, the environment, the scene and the story in their mind in order to understand and extrapolate the contents. However, in many other situations, a 3D scene can enable them to understand the situation or concept and make them familiar with the necessary information. As an example, a 3D scene can allow children to become more interested in the topics they are exposed to. Creating such a 3D scene that represents a story is a time-consuming and challenging process. It involves the creation of objects, their careful placement in the scene to mimic the actual scene and the application of the appropriate material for each object. Moreover, in many situations, as the story evolves, its proper representation requires 3D animation to make it even more complicated. Therefore, there is a need to develop methods to automatically create 3D scenes and 3D animations from a given text.

In this paper, we propose a system to generate 3D scenes from simple text input. We use Natural Language Tool Kit (NLTK) to extract names of objects from the provided text and to identify objects. We also identify the spatial
relations associated with these objects to determine their physical location in
the scene. This paper also provides an object positioning framework to place
objects in a scene by calculating their bounding box values of the objects and
considering the spatial relationships between them, as well as adding motion and
special effects in the established static scene to create animations.

The rest of the paper is organized as follows. Section 2 provides information
about some of the related work. In Sect. 3, we propose and discuss our frame-
work. Section 4 provides a brief overview of the text processing component of
the framework for analyzing the input text for identification and retrieval of key
objects and their spatial relationships. Section 5 provides our object placement
and scene generation approach. Section 6 provides some experimental results
while Sect. 7 provides a few concluding remarks and possible future work.

2 Related Work

To generate a visual representation from provided text, many of the earlier pro-
posed techniques require either the use of annotated objects or pre-formatted
input, like XML. In some cases, such systems may also require the user to “learn
complex manipulation interfaces through which objects are constructed and pre-
cisely positioned within scenes” [3]. The 3D models representing real-world items
may also need to be annotated different set of characteristics to describe their
spatial relationships with other objects. Spatial relations not only play a crucial
role in the placement of objects in a scene but also in describing their associ-
ation with other objects. Prior research assumes that somehow this knowledge
has been provided during the manual annotation process and is accessible.

WordsEye [5] is a system to generate 3D scenes from textual descriptions.
It primarily provides a user a blank slate to represent a picture using words
such that the representation contains both the actions performed by the objects
in the scene as well as their spatial relations. With the help of tagging and
parsing, it essentially converts the provided text into semantic representation,
which can be portrayed by choosing and arranging models from a model database
to form a 3D scene. WordsEye’s authors have continued to improve the system,
with subsequent documents to give more information about how the system
operates. WordsEye consists of a library of approximately 3,000 3D objects and
roughly 10,000 2D images, which are tied to a lexicon of about 15,000 nouns [15].

Put [4] is generally considered as one of the first proposed text-to-scene sys-
tem. It is a language-based interactive system that changes the location of the
objects in a scene by taking input in the form of an expression. Therefore, every
statement needs to be in the form Put(X P Y), where X and Y are the objects
and P is a spatial relationship between them. As an example, one might issue a
command “Put the chair in room at location (x, z)” or “Put the ball in front of
the square table” [4].

In [8], Li et al. also presented a system to generate 3D scenes from text. The
proposed system is based on AutodeskMaya® such that the input text is in the
form of XML. This system consists of three main components: language engine, object database and graphics engine. In the language engine, the user generates an XML file from the provided input text using different tags to describe object names and spatial relationships among them. The system then places the objects in the scene based on their spatial relationships with other objects. The object database contains 3D objects. Based on the requirements of the user input, these objects can be imported into the scene. The graphics engine is responsible of importing and repositioning objects in the scene. To reduce collisions among objects and to prevent overlapping, Lu et al. [10] used the concept of bounding box which itself is an extension to the approach suggested by Li [8]. Chang et al. [3] used natural language descriptions for each 3D scene in their indoor scenes dataset. The system learns from this dataset and recommends a set of suitable 3D scenes according to the spatial constraints and objects provided in the given input text. Depending on the probability of words that are matched between the input text and the description of the scene, the system provides the layouts of a 3D scene in descending order of the probability. An extension of his work is proposed by the same authors as SCEENSEER [2], where the user can iteratively change the location of an object in the selected scene through the input text.

Unlike text to 3D scene generation systems, the text to animation is even more complex. In addition to the challenges that the static scene generation systems face, the text to animation systems also need to recognize the set of actions associated with the objects in the text. Such systems should also be able to assign movement to the object while making sure that the objects neither collide with nor overlap other objects in the scene. The system proposed by Ruqian Lu et al. [11] takes a very limited subset of Chinese natural language as an input which is then transformed into a frame-based semantic description language. The system then constructs a qualitative script consisting of a sequence of scenes with camera movements and lighting for each scene. This qualitative script is then converted into a quantitative script which specifies a series of static image frames to be generated. Finally, the quantitative script is converted to an animation script to get the final animation. CarSim [6] is another system that involves animation. It takes accident reports, written in French, as its input and generates a 3D animation. The system performs linguistic analysis on the text and categorizes the objects into motionless objects, moving objects and collisions. Based on the input text, the system creates animations for accident scenes. Glass et al. [7] system takes annotated fictional text as its input. Constraints like layout, appearance and motion associated with the objects are present in the annotated text. These constraints are converted into mathematical constraints to generate 3D animation. Oshita et al. [12] developed a system that maintains a database of motion clips. For each motion in the input text, one query frame is generated which searches the database for its respective motion clip. Time duration for all of these motion clips are stored in a motion time-table and is used to create the final animation.
3 Framework

To create a visual story from textual narratives, our proposed framework consists of two stages: (i) text processing, and (ii) object positioning as shown on Fig. 1. In the text processing stage, the input, which generally involves a description of a scene or a story, is processed to identify and extract names of object and their spatial relationships. In the second stage, the system retrieves previously identified objects from the system library of 3D objects and imports them into the scene. After importing these objects, the system tries to establish the dimensions of each object (height, width and length) through its bounding box information. In conjunction with the extracted spatial relationships, these dimensions are used to properly place objects in the scene. After placing objects, any motion associated with them is added to the scene by specifying keyframes with respect to time. Unlike previously proposed models, in our framework we specify individual motion functions to add keyframes to each of the object, according to the type of movement indicated in the input text.

Fig. 1. Proposed framework

4 Text Processing

In principle, the user-provided input text in this framework is a description of a scene that contains objects and their spatial relationships. In order to process text input, we use Natural Language Tool Kit (NLTK) in Python (available from https://www.nltk.org/). NLTK is an open-source library that is written in Python and runs on all operating systems (Windows, MAC OS and Linux) and is supported by Python. It includes both symbolic and statistical natural language. It also provides easy-to-use interfaces to over 50 annotated corpora and lexical resources along with a suite of text processing libraries for classification, tokenization, parsing, and semantic reasoning [9]. Furthermore, it contains implementations of many different algorithms to provide various different functionalities and are used in many research implementations and projects.
These algorithms are trained on a structured set of texts. Using NLTK, every word in the text is tagged such that the input text is first split into sentences and then to individual words. The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech (POS) tagging. With the help of these tags, nouns and positions, i.e., the object names and the inter-object spatial relationships are extracted. A simple example of text processing for basic 3D objects is shown on Fig. 2. The final outcome of text processing is a list containing the names of objects that are present in the scene to be created, as well as their spatial relationships.

```python
text = "Sphere on cube and cone on sphere"

sentence_tokenized = nltk.sent_tokenize(text)  # splitting the input text into sentences
  ⇒ ['Sphere on cube', 'cone on sphere']

word_tokenized = nltk.word_tokenize(sentence_tokenized)  # splitting each sentence into words
  ⇒ [['Sphere', 'on', 'cube'], ['cone', 'on', 'sphere']]

parts_of_speech_tagging = nltk.pos_tag(word_tokenized)  # tagging parts of speech to every word
  ⇒ [('sphere', 'NN'), ('on', 'IN'), ('cube', 'NN')]
  ⇒ [('cone', 'NN'), ('on', 'IN'), ('sphere', 'NN')]

  // where NN stands for noun and IN stands for preposition

Fig. 2. Example of text processing
```

5 Object Positioning

In this section, we discuss the suggested approach to use output from the text processing stage to generate a 3D scene.

5.1 Bounding Box

In a 3D animation software, a bounding box stores the values of $X_{min}$, $Y_{min}$, $Z_{min}$, $X_{max}$, $Y_{max}$ and $Z_{max}$ of a 3D object, with respect to the $X$, $Y$ and $Z$-axis. These values provide us not only the physical dimensions of a 3D object but allows us to prevent overlapping of objects during their placement in the scene. For example, the dimensions (length, height and width) of a table object in 3D space can be found by simply using $X_{max} - X_{min}$, $Y_{max} - Y_{min}$ and $Z_{max} - Z_{min}$, respectively (Fig. 3).
5.2 Spatial Relationships

Spatial relationships provide information about the position of two or more objects in space, relative to oneself and to each other. Spatial relationships provide the ability to understand locations and directions through specific words. In text-to-scene generation systems, these words are vital to specify both the position of an object and its relationships to other objects in the scene. Table 1 shows a few of the different spatial relationships that are supported by our system. We have classified spatial relationships in two categories: location and direction. This classification is based on the distance at which an object is placed relative to the position of another object in the scene. For the location spatial relationships, one face of the object is in contact with another object’s face and the location spatial relationships are on, under, in front of, left, right, inside, behind, above and below. For directional spatial relationships, an object is placed at a certain distance from another one and the direction spatial relationships are east, west, north, south, southeast, southwest, northeast, and northwest.

| Location-based spatial relationship | Direction-based spatial relationship |
|-----------------------------------|------------------------------------|
| on, under, in front of, left, right, behind | east, west, north, south, southeast, southwest, northeast, northwest |

6 Scene Generation

_Autodesk Maya®_ is a very popular and a leading 3D computer graphics application software that is capable of handling complete animation production pipeline and runs on all three major platforms, i.e., Microsoft Windows, MacOS and Linux. It is a 3D modeling and animation software and has very strong capability to generate realistic special effects and high quality rendering through a
set of menus. It provides C++ Application Programmers Interface (API) and also allows to extend the functionality by providing access to Maya Embedded Language (MEL), as well as Python. With the help of MEL or Python, users are able to write a script or develop a plugin to accommodate repetitive tasks by running them within Maya. We have used Maya to generate 3D scenes. With the output from text processing stage containing names of objects and their spatial relationships, the system imports these objects from its own object library. These objects can be resized and their visual characteristics such as colours or texture can be changed to reflect the resemblance with that of the objects in the textual story, either before or after generation of the final scene. New objects can be added to the library as well as existing objects can be replaced with the new ones. During the process of scene generation, if the object is not present in the library, the user gets a choice to either add a new object to the library or continue the scene generation without that object. The objects are initially imported into the scene and placed at the origin, i.e., position (0, 0, 0) in the 3D space. Thereafter, by using the bounding box, the system extracts the dimension \((X_{\text{min}}, Y_{\text{min}}, Z_{\text{min}}, X_{\text{max}}, Y_{\text{max}} \text{ and } Z_{\text{max}})\). Depending on the spatial relationships between the objects, the system calculates either 1D or 2D relationship among length, height and width, since every spatial relationship may not require all the three dimensions. For example, the location spatial relationship ‘on’ involves the height of the object, while the direction spatial relationship ‘northeast’ involves both length and depth of the objects to place them in the scene. Figure 4 shows different object locations with respect to different spatial relationships.

The next step after object positioning is to add animation and any special effects to the objects. Autodesk Maya allows different animation techniques to
add motion to an object. The most common among them is the keyframe animation, in which the object’s spatial location as well as key attributes are saved with respect to the time. We have movements with different characteristics that can be assigned to any of the objects in the scene. However, it is important to note that not all objects will have exact or similar movements. As an example, consider a scene a walking person and a moving car. In a realistic scenario, the walk will be much slower than the speed of movement of the car. Consequently, the distance traveled by both for the same period of time will be different and as such should be indicated in the animation. Finally, the system adds any special

**Fig. 5.** In front of the table there is a chair. There is a book on the table. Left of the book is a vase with flowers in it. Northeast of the book is a lamp.

**Fig. 6.** There is a car on road and car is moving. There is a walkway on the left of the road. There is a building to the left of the footpath. The building is facing towards footpath. The building is on fire.
effects that are associated with the objects or the scene. Figure 5 is an example of a 3D scene generated using our framework for the input text as shown in the figure (Figs. 6, 7, 8 and 9).

![Image 1]

**Fig. 7.** During sunrise, the sky is full of clouds and birds are flying above the mountains.

![Image 2]

**Fig. 8.** There is a table inside a room and a couch in front of the table with a Stool right to the couch. There is a lamp on the stool and a vase on the table and flowers in the vase. Northeast of the table there is a plant. Northwest of table there is a bookcase. A carpet is on the floor. A clock is on the front wall and a TV below the clock. A photo frame is on the left wall.
Fig. 9. Odette building is on the right of the Sunset Avenue and the Erie hall is on the left of the Sunset Avenue. Next to the Erie Hall, there is Lambton Tower and Chrysler Hall is north of the Lambton Tower. North of the Odette Building is Toldo Building. Students are walking on the Sunset avenue.

7 Conclusion and Future Work

In this paper, we have proposed a new framework to generate a 3D scene from a given text. Our method uses the Natural Language Tool Kit library and Python to extract the names of objects and their spatial relationships from the input text. These objects are imported from our database and are then placed and repositioned in the generated 3D scene based on their spatial relationships and dimensions. The latter are calculated using bounding box values. Finally, animations and special effects are added to the objects to make the generated scene dynamic, more realistic and complete.

The overall system can be improved in many ways. In particular, there is a clear need for better handling of the input text. One can develop more NLP libraries in Python within Maya to perform semantic analysis of complex input texts. In its current form, it is capable of processing smaller sentences which is not suitable for all types of stories or situations. There is also a need to extract information from relatively larger passages to build a scene.

References

1. Anagnostopoulos, C.N., Anagnostopoulos, I., Loumos, V., Kayafas, E.: The primary care PTSD screen (PC-PTSD): development and operating characteristics. Primary Care Psychiatry 9(1), 9–14 (2003)
2. Chang, A.X., Eric, M., Savva, M., Manning, C.D.: SceneSeer: 3D scene design with natural language. ArXiv abs/1703.00050 (2017)
3. Chang, A.X., Savva, M., Manning, C.D.: Interactive learning of spatial knowledge for text to 3D scene generation. In: Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces, pp. 14–21 (2014)
4. Clay, S.R., Wilhelms, J.P.: Put: language-based interactive manipulation of objects. IEEE Comput. Graph. Appl. 16(2), 31–39 (1996)

5. Coyne, B., Sproat, R.: WordsEye: an automatic text-to-scene conversion system. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 487–496 (2001)

6. Dupuy, S., Egges, A., Legendre, V., Nugues, P.: Generating a 3D simulation of a car accident from a written description in natural language: the CarSim system. In: Proceedings of the Workshop on Temporal and Spatial Information Processing, vol. 13, pp. 1–8 (2001)

7. Glass, K.R., Bangay, S.: Automating the creation of 3D animation from annotated fiction text. In: Proceedings of the IADIS International Conference on Computer Graphics and Visualization, pp. 3–10 (2008)

8. Li, C., Yin, C., Lu, J., Ma, L.: Automatic 3D scene generation based on Maya. In: Proceedings of 2009 IEEE 10th International Conference on Computer-Aided Industrial Design Conceptual Design, pp. 981–985 (2009)

9. Loper, E., Bird, S.: NLTK: the natural language toolkit. ArXiv arXiv preprint cs/0205028 (2002)

10. Lu, J., Li, C., Yin, C., Ma, L.: A new framework for automatic 3D scene construction from text description. In: Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing, vol. 2, pp. 964–968 (2010)

11. Lu, R., Zhang, S.: Automatic generation of computer animation: using AI for movie animation (2002)

12. Oshita, M.: Generating animation from natural language texts and semantic analysis for motion search and scheduling. Vis. Comput. 26(5), 339–352 (2010)

13. Rizzo, A., et al.: A virtual reality exposure therapy application for Iraq war military personnel with post traumatic stress disorder: from training to toy to treatment. In: NATO Advanced Research Workshop on Novel Approaches to the Diagnosis and Treatment of Posttraumatic Stress Disorder (2006)

14. Roy, M.J., Rizzo, A., Difede, J., Rothbaum, B.O.: Virtual reality exposure therapy for PTSD (2016)

15. Ulinski, M., Coyne, B., Hirschberg, J.: Evaluating the WordsEye text-to-scene system: imaginative and realistic sentences. In: Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018