

University of Windsor

## Scholarship at UWindor

---

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

---

10-18-2019

### Dynamic Scene Creation from Text

Havish Kadiyala  
*University of Windsor*

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

---

#### Recommended Citation

Kadiyala, Havish, "Dynamic Scene Creation from Text" (2019). *Electronic Theses and Dissertations*. 8143.  
<https://scholar.uwindsor.ca/etd/8143>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.

**Dynamic Scene Creation from Text**

By

**Havish Kadiyala**

A Thesis

Submitted to the Faculty of Graduate Studies  
through the School of Computer Science  
in Partial Fulfillment of the Requirements for  
the Degree of Master of Science  
at the University of Windsor

Windsor, Ontario, Canada

2019

© 2019 Havish Kadiyala

# **Dynamic Scene Creation from Text**

by

**Havish Kadiyala**

APPROVED BY:

---

S. Towson

Department of Psychology

---

D. Wu

School of Computer Science

---

I. Ahmad, Advisor

School of Computer Science

October 16, 2019

## DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

## ABSTRACT

Visual information is an integral part of our daily life. Typically, it tends to convey more information than simple textual information. A visual depiction of a textual story, as an animation or video, provides a more engaging and realistic experience and can be used in different applications. Examples of such applications include but are not limited to education, advertisement, crime scene investigation, forensic analysis of a crime, treatment of different types of mental and psychological disorders, etc. Manual 3D scene creation is a time-consuming process and requires expertise of individuals familiar with the content creation environment. Automatic scene generation using textual description and a library of developed components offers a quick and easy alternative for manual scene representation and proof of concept ideas. In this thesis, we propose a scheme for extraction of objects of interest and their spatial relationships from a user-provided textual description to create a 3D dynamic scene and animation to make it more realistic.

**Keywords:** text to scene generation, 3D animation, multimedia, computer graphics, spatial relations, bounding box

## DEDICATION

*I dedicate this thesis to my beloved parents Mr. Krishna Vara Prasad Kadiyala and Mrs. Yugandhari Kadiyala, and to the rest of my family and friends.*

## ACKNOWLEDGEMENTS

First and foremost, I would like to express profound thankfulness to my supervisor, Dr. Imran Ahmad, who has supported me throughout my thesis with his knowledge and expertise in this exciting field of research. His ideas and suggestions have helped me become more creative, without which I would not have been able to complete this research.

I would like to offer sincere gratitude to the advisory group members, Dr. Dan Wu and Dr. Shelagh Towson for their significant remarks and recommendations for my research.

I would like to thank all my friends, who have supported and helped me throughout my studies. I also thank my parents and family for their blessings and financial support which enabled me to complete my studies successfully.

## TABLE OF CONTENTS

DECLARATION OF ORIGINALITY .....	iii
ABSTRACT .....	iv
DEDICATION .....	v
ACKNOWLEDGEMENTS .....	vi
LIST OF TABLES .....	ix
LIST OF FIGURES .....	x
LIST OF ACRONYMS .....	xi
<b>CHAPTER 1: INTRODUCTION</b> .....	<b>1</b>
1.1 Applications .....	2
1.1.1 Entertainment.....	2
1.1.2 Scientific.....	4
1.1.3 Other Industries .....	5
1.2 Motivation .....	7
1.3 Problem Statement .....	8
1.4 Thesis Contribution.....	9
1.5 Thesis Organization.....	9
<b>CHAPTER 2: 3D ANIMATION PIPELINE</b> .....	<b>10</b>
2.1 Preproduction .....	11
2.2 Production .....	15
2.3 Postproduction.....	19
<b>CHAPTER 3: LITERATURE REVIEW</b> .....	<b>21</b>
3.1 Text to Scene.....	21
3.2 Text to animation .....	27
<b>CHAPTER 4: AUTODESK MAYA</b> .....	<b>33</b>



4.1 Maya Architecture.....	33
4.2 Python 2 vs 3.....	35
4.3 Grounding Language.....	35
<b>CHAPTER 5: METHODOLOGY .....</b>	<b>37</b>
5.1 Bounding Box .....	37
5.2 Spatial Relations.....	38
5.3 Framework .....	39
5.4 Algorithm .....	40
5.5 Scene generation .....	41
5.5.1 Text Processing.....	41
5.5.2 Object positioning.....	43
5.6 Results .....	46
5.6.1 Tools .....	46
5.6.2 Generated 3D scenes .....	47
5.7 Discussion .....	52
5.8 Conclusion.....	53
5.9 Future work .....	54
<b>REFERENCES.....</b>	<b>56</b>
<b>VITA AUCTORIS .....</b>	<b>60</b>

## LIST OF TABLES

Table 1: Pros and Cons of different programming languages in Maya [34] .....	34
Table 2: Spatial Relations supported by different systems .....	38
Table 3: Spatial Relations supported by our system .....	39
Table 4: List of tools used for the implementation of our proposed system .....	46
Table 5: Comparison between different 3D scene generation systems .....	53

## LIST OF FIGURES

Figure 1 Indoor and outdoor rendering for architecture .....	5
Figure 2 Animation of gunshots from a specific location .....	6
Figure 3 Example of Augmented reality .....	7
Figure 4: 3D animation pipeline .....	10
Figure 5: Example of a storyboard .....	14
Figure 6: A simple character art in different views .....	15
Figure 7: Final model before textures (left) and after applying textures (right) ....	17
Figure 8: Rigged character .....	18
Figure 9: The daisy is in the test tube .....	22
Figure 10: The bird is in the birdcage. The birdcage is on the chair. ....	22
Figure 11: Example of XML file as input .....	24
Figure 12: Generated 3D scene .....	25
Figure 13: Scene generated using four different methods .....	27
Figure 14: CARSIM system input(left) and output(right) .....	30
Figure 15: Annotated fiction text .....	31
Figure 16: Example of abstract constraints .....	32
Figure 17: Example of Oshita's system .....	32
Figure 18: Maya Architecture .....	33
Figure 19: Different versions of Visual Studio installed inside Maya and system.	36
Figure 20: Table in 3D space with respect to X, Y, Z axis.....	37
Figure 21: System framework.....	40
Figure 22: Representation of how objects are connected with the spatial relations and its respective final scene.....	42
Figure 23: Spatial relations with respect to a table .....	44
Figure 24: A Generated 3D scene with fire effect .....	46
Figure 25: Generated 3D scene for input text 1 .....	47
Figure 26: Generated 3D scene for input text 2.....	48
Figure 27: Generated 3D scene for input text 3.....	49
Figure 28: Generated 3D scene for input text 4.....	50
Figure 29: Generated 3D scene for input text 5.....	51
Figure 30: Tasks achieved in production stage by using our framework [3].....	54

## LIST OF ACRONYMS

MEL	Maya Embedded Language
NLP	Natural Language Processing
API	Application Programming Interface
VFX	Visual Effects
GUI	Graphical User Interface
CAD	Computer-Aided Design
PTSD	Post-Traumatic Stress Disorder
XML	Extended Markup Language
NLTK	Natural Language Tool Kit

## CHAPTER 1: INTRODUCTION

Real-world interactions take place in all three dimensions of space. Objects in the real world are 3-dimensional as they have height, width and depth while a drawing on paper is often 2D and has only height and width. 2D images describe the scene from a particular viewpoint, while a single 3D scene describes the real world from different viewpoints. 3D models are more realistic than 2D. 3D scenes can be used to create 2D images or drawings directly in different viewpoints. 3D modelling is used in different industries like films, video games, advertising, architecture and virtual reality.

In order to explain a story or a situation, we use natural language as a medium of communication. The person who is listening to this story visualizes the objects and environment in story and understands the content. For example, a 3D scene can be used as a visual aid for teaching children, which may make them interested in the topic and visualize the actual story that is being taught by the teacher. Creating such a 3D scene that represents a story is a time-consuming and challenging process and involves creating objects, choosing materials for each object and carefully placing them in the scene. Therefore, there is a requirement for automatic text to scene generation methods that could take text as input and generates a 3D scene accordingly.

Creating digital visual information from language or story/text is attained by describing the real world. Much of the development in 3D scene generation framework is primarily based on the initial work of PUT [1] in 1996. Due to the limitations in technology during that time, authors of this work were limited to simple grammar, pre-formatted input and placement of objects in a scene. However, in recent years, the technology has improved significantly in fields like graphics, computer vision and Natural Language Processing (NLP). This technological growth has enabled us to extract information from the input text more accurately and create more realistic 3D scenes and animations.

The objective of this thesis is to create a system that is capable of generating a 3D scene from an input text, describing the objects, their placement, position and their mutual relationships. To generate 3D scenes from text, we extract the names of objects and their

spatial relations from input text that describes the real world entities; then objects are imported from the local object library. The object library consists of individual 3D object files, most of which were created by the author, with a few downloaded from the multimedia websites that sell 3D objects. These companies sell 3D objects under their terms and conditions where any kind of redistribution is prohibited. Based on the objects that are mentioned in the input text, the 3D objects are imported into the final scene. After importing the objects, based on the spatial relations, objects are rearranged and organized. Finally, motion to the objects and effects are added to the scene according to the information provided in the input text.

## **1.1 Applications**

Generated 3D scenes can be used in many ways (e.g., video games, advertising, architecture, law) in the animation industries, which can be classified into Entertainment, Scientific and Other Industries. Each of these industries has multiple subcategories.

### **1.1.1 Entertainment**

The primary goal of the entertainment industry is to provide entertainment to the audience in the form of 3D animated films, television shows, video games and advertising commercials to name a few.

#### **Film**

Fully animated film and visual effects film are the two types of films that are created in the 3D animation industry. In fully animated films, all of the visual components on screen are generated using 3D animation software and rendered. Examples of fully animated films include Toy Story, Incredibles, Minions, Inside Out, etc. Visual effects films are shot with real actors, but the backgrounds and other effects are computer-generated by using a 3D animation software. Jurassic Park, Avengers, Avatar, etc. are examples of films with 3D visual effects. Depending on the complexity and length of the film, it may take up to four years to create a film and the production team can be fairly large. As an example, more than 300 professionals were involved in the production of the animated movie Ratatouille.

Unlike fully animated feature films, visual effects films are shot by a regular movie crew. The shots that are completed are sent to visual effects studios to add required effects to the shots.

### **Television**

Creating a 3D animated television show is both expensive and time-consuming. Artists need to have creative thoughts to make the story more engaging for the audience. Several shows are being designed with 3D software such as Stinky and Dirty, Mickey Mouse, Sponge Bob Square Pants, etc. A common usage of 3D animation in the television industry is the addition of 3D visualizations to regular shows on channels such as Discovery Health Channel, History Channel, and Science Channel. These visualizations are used in the educational shows to help the audience understand specific topics. The TV industry with a small budget and a short amount of time, has to create different seasons and in each season they have to create several episodes, while the film industry works on a particular story that has a total runtime around 120 minutes has an advantage of both money and time.

### **Video Games**

In the video game industry, artists use 3D graphics software to create virtual worlds and characters that are used in a video game engine [2]. This industry is massively successful and is at least as profitable as the film industry. There are two main fields in the video game industry: in-game 3D animation, which creates the actual game environment that players are involved in while playing the video game, and game cinematics, which are created cinematically and used as story scenes to help drive the story forward between the game levels. Game cinematics, like movies, are restricted today only by the budget and time required to design the 3D animation scenes and to render the final scenes to be played in the video game. Game cinematic artists do similar kind of work as the film 3D animators but typically by following a much faster timeline. In-game cinematic scenes and cinematic game trailers are of high quality that can compete in the film industry.

### **Advertising**

Advertisements that are telecasted on television channels and on web pages (print ads, and ads that play before and after a video on YouTube) are typically short and take 10 seconds to four or five minutes to show or describe a product in detail. These short animations must

be able to present a great deal of information in a short period of time. For product advertising, an artist creates a 3D model to serve as a prototype of an actual product to create consumer interest in that product.

Advertisements have a very high level of quality about the products and are created in a shorter amount of time. Advertising animation studios are medium-sized and follow a robust workflow to provide faster outputs for this type of animation [3].

### **1.1.2 Scientific**

The scientific industry primarily uses 3D animation in the field of medicine and architecture.

#### **Medicine**

In the medical industry, 3D animation is used in many ways, from creating a visualization of a specific medical event to describing a biological reaction. The most popular medical usage of 3D animation type is in medical visualization for education or marketing. Animation is used to teach the public and medical staff about new techniques or drugs. 3D animation can create a rich visual guide to human and biological systems and can provide important information in a short amount of time.

Ongoing studies are examining how video games are used to help heal brain injuries [4]. These video games stimulate different regions of brain, potentially helping the regrowth of brain tissue. These studies are fairly new but are still showing good results, which means that more of these types of games have to be created for other healing applications [4]. 3D animation in the medical industry is a vastly growing market that can be profitable to a single artist or small studio of professionals. A huge drawback in this industry is that most people training today in 3D animation would prefer to work in video gaming industries or film industries and not for a drug company or university research project as the work in gaming companies is more creative and interesting while the work in a drug company is to create an animation that resembles a product [3].

#### **Architecture**

Architects are using 3D animation to create better and more stable computer-aided designs. Today architects use CAD [5] programs not only to build models but also to test and



visualize those models to see how constructions would look like before they are constructed. Software such as Autodesk AutoCAD [6] and Autodesk Revit [7] allows architects to test the endurance of designs under certain circumstances, to see whether they can withstand a specific type of natural disaster. The CAD files can be converted and then rendered in 3D animation software such as Autodesk 3ds Max and Autodesk Maya to enable clients to see what a structure could look like from inside and outside. This type of work is becoming more popular and can be a very cost-effective way to test particular material looks of a building before actually constructing it. An example of interior and exterior architectural rendering is shown in Figure 1.



*Figure 1 Indoor and outdoor rendering for architecture [3]*

### **1.1.3 Other Industries**

#### **Law**

In Law, 3D animation can be used for forensics and real accident scene reconstruction and simulation. This kind of animation is created to prove, disprove, or elaborate on evidence in a court case, to help either the defense or the prosecution. It can have pure computer simulations or just a key framed animation of the crime scene to assist the judge or jury to study the crime scene if required. It can be used as an example, to prove that a criminal could or could not have shot someone from a particular location or to describe a car accident situation. These types of animations are not used as clear evidence but can be used to describe the scene in which the incident happened.

3D animation can also be used to create a crime scene by scanning it using a 3D laser scanner [8]. This 3D laser scanning can recreate a crime scene perfectly so that it can be used as a reference when needed. This 3D scan can be accurate to millimeters and so it can

be crucial to a court case or an investigation. These 3D scanners are too expensive for an individual person who wants to use them.



*Figure 2 Animation of gunshots from a specific location [3]*

The technology driving the 3D animation industry is changing every year. The new 3D animation fields are augmented reality and projection mapping.

In Augmented Reality(AR), a user looks at the real world and can add 3D elements to it. Typically, we would look through a camera and can place 3D objects though the camera as seen in Figure 3. Even the real-world objects using AR can be measured. Other viewing devices, are head-mounted with a see-through visor that adds the 3D elements to the real world.

Projection mapping is a new method that can transform any surface, into a video display, typically large buildings. This technique uses projectors to project a 3D animation onto a building. It displays new and exciting effects such as building destruction or fire on the surface. This technique is used to create many exciting effects, and it may become one of the major applications in 3D animation in the future.



*Figure 3 Example of Augmented reality [3]*

## **1.2 Motivation**

The tendency of humans to understand videos or images involves more than understanding the textual content. Animation is created by moving the still pictures at a certain number of frames per second in order to create the illusion that the object is moving. We have seen different applications where 3D animation can be used. It can also be used as a visual aid to teach children so that they can visualize the actual story that is being taught by the teacher. In the medical field, Post-traumatic stress disorder (PTSD) [9] is a mental illness which is caused by a terrifying event like warfare or traffic collisions. After facing such conditions directly, the person can have symptoms like losing interest in talking with others, having bad dreams, avoiding people, difficulty in concentrating etc. To treat these kind of patients, health practitioners can recreate the event in a virtual environment and analyze the health parameters of the patient when exposed to it. Thus, a software that helps in converting a text or speech into a realistic virtual scene can be highly beneficial for health practitioners in treating people suffering from PTSD.

The aim of generating 3D scenes from text has many challenging tasks associated with it. The system has to be able to identify the objects and their spatial relations in the text and also identify how the objects are related to one another in the scene according to their spatial relations. The system should also be able to know the dimensions of the objects to place them in scene with respect to their spatial relations. These tasks indicate that the text to 3D scene generation is a broad and challenging task.

In this thesis, we describe how language can be linked to represent geometric representations of objects with the help of spatial relations between them which are key elements in describing the real world.

### **1.3 Problem Statement**

A 3D scene is required by many fields in different industries. Creating a 3D scene is time-consuming and requires a lot of effort to learn complicated 3D animation software. To reduce human effort in learning the software and creating 3D scene, we propose a framework through the Autodesk Maya to generate 3D scenes from input text to describe a scene in the real world.

There are several key tasks in generating a 3D scene from text descriptions.

- Determining the key objects to be present in the scene.
- Determining the spatial relations that are connecting the objects.
- Placement of objects in a scene by finding the dimensions of the objects.
- Adding motion to the objects.
- Adding effects to individual objects and to the whole scene.

We explain each of these tasks in the following chapters, which lead us to the successful creation of a 3D scene from text descriptions.

## **1.4 Thesis Contribution**

In this thesis, we make the following contributions.

- The proposed system generates 3D scenes from the input text. We used Natural Language Tool Kit (NLTK) to extract objects names from the input text to identify the objects that are to be present in the scene.
- We extract different spatial relations that are associated with the objects from the input text to decide their location in the scene.
- We provide an object positioning framework that helps to place the objects in a scene by calculating the bounding box values of the objects and considering the spatial relations between them.
- We also add motion and effects to the static scenes.

## **1.5 Thesis Organization**

The rest of the thesis is organized as follows: Chapter 2 explains the 3D animation pipeline. Chapter 3 provides a review of previous works that is done on the text to scene and text to animation generation. Chapter 4 describes Autodesk Maya and discusses extending Maya's functionality through scripting and/or plugins. Chapter 5 introduces the proposed framework, our approach for text to scene creation and results. Finally, in chapter 6 we conclude with some discussion and suggestions for future work in text to scene generation.

## CHAPTER 2: 3D ANIMATION PIPELINE

A 3D animation pipeline consists of a combination of people, hardware and software, to design a 3D animation product or asset that works in a particular order where the final product could be a feature film, short film, television show, or video game. Depending on the project type, the particular software and hardware may change, but the fundamental stages of the 3D animation is similar. The three main stages of 3D animation pipeline are as follows:

- Preproduction
- Production
- Postproduction

In the production stage, the 3D artists design the product and in postproduction, the artists deliver the project in its final form. The entire 3D animation production pipeline is shown in Figure 4.

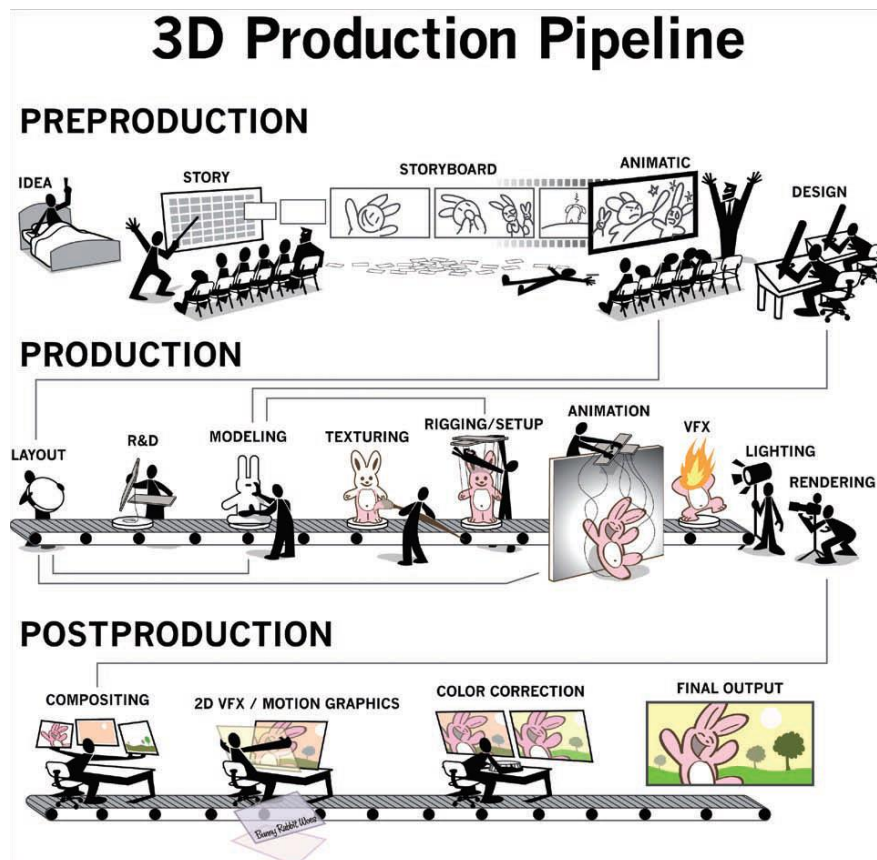


Figure 4: 3D animation pipeline [3]

## **2.1 Preproduction**

Preproduction is the planning, designing, and research stage of the complete 3D project. This is an essential stage because it is where the ideas are generated and production plans are designed to manage the project. A great idea with a solid production plan has a much better possibility of being completed than a great idea with no plan.

The preproduction team is divided into two groups: artists/story writers and management. Artists/storywriters create ideas, story, and designs, and the management group creates a production plan. In addition, a creative management group works directly with the artists/storywriters. They are also in charge of selling the project to investors.

All components of the diverse entertainment industry handle the preproduction stage in very much the same way. Artists use this stage to get good ideas, writing those ideas into an engaging story and then deciding how to tell that story visually. These industries may end up using approximately half of their entire production timeline in the preproduction stage.

After watching a movie which is adapted from a best-selling book, one might have heard or even stated, “The book was way better than the movie.” That is because there is a significant difference between the storytelling of a written story, since the former is intended to be read, whereas the latter is the storytelling in visual story, which is intended to be seen. To make a story expressed well onscreen, the entire production team must consider different aspects: to convey to the audience what is going to happen with different camera angles; choosing of colour in the project to manage the audience’s mood and feeling. To further influence the mood of the audience, the music, sound, or silence should be handled along with the story.

In the architecture industry, 3D artists spend most of their production time in the preproduction stage. Design and functionality are the main goals of these artists. If the architecture project is completely approved, the end product is an object to be created in the real world. Artists working in product visualization spend much time in this preproduction phase to ensure that their product design is just the way they want it. The product can be viewed in the physical world before putting it into production by using a



rapid prototype for the final model. Rapid prototyping is creating a virtual object in the real world by using additive construction technology like a 3D printer which has an intense laser that fuses small particles of plastic together to make an object.

In the medical-animation industry, the artists spend their time in research and development on the type of animation they need to produce. Artists should be very clear about the type of medical technique or organic system that they need to describe in their project visually. During this time, the artists engage in extensive research and use expert consultants to validate the factual correctness of their visualizations. As in other subfields of the entertainment industry, these artists also have to understand how to best display what they are trying to explain or describe. They must consider matters varying from the correct placing of cameras to the overall animation and the final product.

Unlike others, the legal industry may not spend much time in the preproduction stage. Teams from these industries make use of 3D computer simulation in order to recreate any crime incident to check the possibilities of how the crime can be happened. A forensics team might employ a 3D scanner to infer data from a crime scene for an investigator to be able to go back into the crime scene “virtually” much later after the crime. Artists in this industry should research how best they can depict the scene virtually rather than concentrating on outlines of an object. The more basic and general the designs of objects, the better in this industry, because inaccurate detail could mislead the investigator in their decision-making process. The preproduction stage consists of five components as follows:

- Idea
- Story
- Storyboard
- Animatic
- Design

### **Idea**

An idea for a project can come from anywhere and from almost anything. It can be sparked by a word, a sentence, a smell, a colour, a sound, passing conversation with a stranger, or eavesdropping on someone else's conversation. That spark is enough to ignite a dialogue



within yourself or with others who want to work out that idea. After having an idea, a minimum amount of effort is needed to verify the idea by sharing with others and by yourself so that we can know the difficulties that might arise in future. By having a commitment towards the idea, we should work properly to make the idea into reality. After having a good idea, we should figure out the different characters to be part of this project, required human and computing resources for this project.

### **Story**

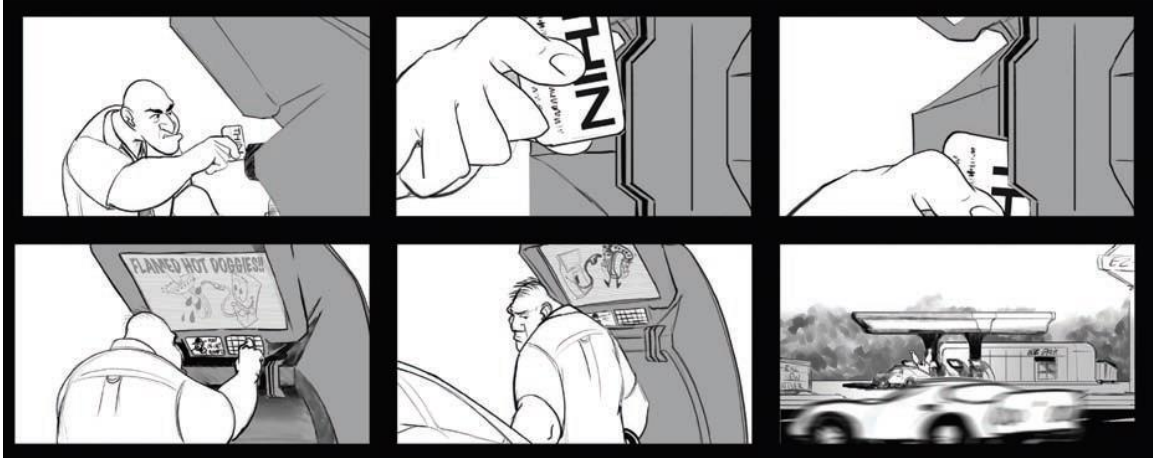
A script or screenplay is the formal written form of the final story. It is written with the basic character movements, environment, time, actions, and dialogues. This script is meant for the preproduction and production team to create a visual idea of the whole story. Many people from those teams gather information from it quickly.

A script is the backbone for the entire project. By seeing it the director or any team members can know what they have to do next, and they can even verify that the work done up till that point is consistent with the story.

### **Storyboard**

A storyboard is the visual form of the entire story. It contains early ideas of camera staging and early designs of potential visual effects. Each image in a storyboard visually describes an essential scene in the story from a script. Figure 5 shows an example of a storyboard. A storyboard reveals gaps and continuity holes in the story that need to be fixed.

A storyboard is used by the entire production team to understand what is going on in the project visually. The storyboard is like a blueprint for an animation project that describes how the scenes should look before starting the project. Industries like product visualization and architecture use storyboards to show the way that a product should best describe or the best way to show the building infrastructure.



*Figure 5: Example of a storyboard [3]*

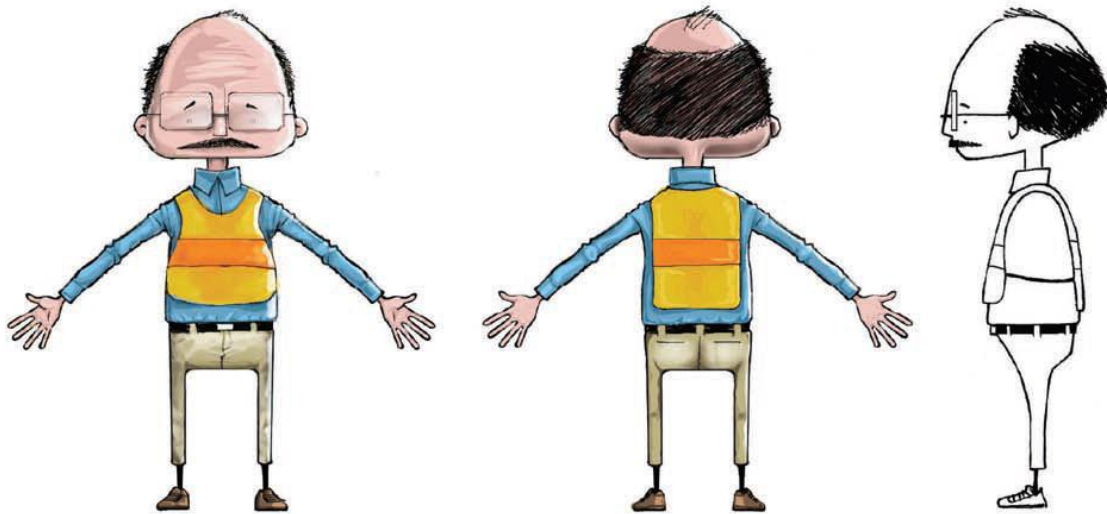
### **Animatic**

An animatic is a short animation of the entire story. It is created by seeing the storyboard images with short dialogue and simple sound effects, to show the flow of the project. The preproduction artists go back and forth among the storyboards and animatic to create the final edit of the whole project.

The animatic enables the director and editor to create the editing style and pacing for the entire project. It may seem that the project is completely finished when we see the individual shots, but when we put them together according to the script, they may not run visually. Animatic is used to pre-visualize visual effects in live-action films. The producers of these projects like to know what all of the shots will look like before they actually start shooting. This helps the director and director of photography in knowing how the scene has to be shot to match the visual effects. It also helps the actors understand how they are going to interact with the virtual character in the movie that is not actually present in the real world.

### **Design**

In this component, the final look of the characters, prop design, clothes, and environment designs are decided. An artist can draw many variants for character design in a day and then finalize a design; a professional 3D artist needs at least a week to create a final model. We will get to know what the character will look like and can make changes if required before creating it on a 3D animation software.



*Figure 6: A simple character art in different views [3]*

## **2.2 Production**

In the production stage, all the scripts, storyboards and designs are given to the appropriate artists to make an animation. The production stage becomes much easier for creating the project if the preproduction stage is done well. As many problems should be solved and design decisions made as possible during the preproduction stage, so that production artists do not have to guess at what they are supposed to be creating. The production stage includes the following components:

- Layout
- Research and Development(R&D)
- Modelling
- Texturing
- Rigging/Setup
- Animation
- Visual Effects(VFX)
- Lighting and Rendering

### **Layout**

The shape and size of the object are defined in this stage. From the pre-production stage, the layout artist takes necessary information such as the object's shape and size and begins

simple animation of the objects. In this stage, the models are not designed to include details like a face or fingers. Basic animation transformation information showing a character moving from one point to another, or the direction that the character is facing to the camera, is all that is required. The basic animation allows the director to start designing the composition of each shot from the camera's point of view. The 3D layout will also grow with the rest of production as the production moves forward. The 3D layout stage allows other components of the production stage to begin working sooner than they usually would in the pipeline.

### **Research and Development**

The research and development component traverses through the entire 3D animation pipeline. Artists from different components sit together with technical directors to discuss upcoming technical challenges in the project. For example, in Pixar's Finding Nemo movie, the Research & Development team had to create the look of water, including the objects that present inside water like plants and fishes. At the time of that film's production, no 3D animated film had shown water, as it was challenging to control and render efficiently. Yet Finding Nemo [10] has great animation shots inside water in almost every shot. In DreamWorks Animation's Monsters vs. Aliens [11], the R&D team had to create a character B.O.B., a blue gelatinous character and control it. It is very challenging to create a character that had no real shape, but that has to be shaped when needed. R&D teams are working on the next best thing and creating effects that we haven't seen yet.

### **Modelling**

A model is a geometric surface that represents an object which can be created, rotated and viewed in a 3D-animation software such as Autodesk's Maya [12], 3ds Max [13], or Blender [14]. A model can also be created by scanning a real object using a scanner and creating a 3D representation of that object in a 3D-animation software.

There are different types of modelling software in the 3D animation industries, and each of them is used for specific purposes. The entertainment industries use Autodesk's Maya, 3ds Max, or Blender. These softwares can do the entire production stage: modelling, animation, physical dynamic simulation, texturing, lighting and rendering. The architecture

and product-visualization industries use Autodesk's AutoCAD, the Robert McNeel & Associates Rhinoceros and Dassault Systemes SolidWorks products, and Google SketchUp Pro for creating objects that are specific to their industry. These softwares cannot do animation, dynamics, and rendering, but do an excellent job of exact-scale production modelling of objects. Law and medical industries can use any of the above software, depending on their requirements for a specific project. After the models are completely designed, the models are sent to the texture artist to be painted and shaded.

### **Texturing**

Texture artists apply colour and surface characteristics to the geometric models in the texturing component. The 3D models generally come to the texture artist in a 3D-animation software's default colour. The texture artist's work is to make the model's exterior look like its real-world object. For example, if the model is a wood table, the artist will make sure that the table when rendered looks like the wood it would be made of.

Texture artists use different softwares and techniques to complete their work. The artist might hand-paint those textures or use images to add texture to an object. Figure 7 shows an object before texturing in the default gray colour of 3D software, and the same object after textures are painted.

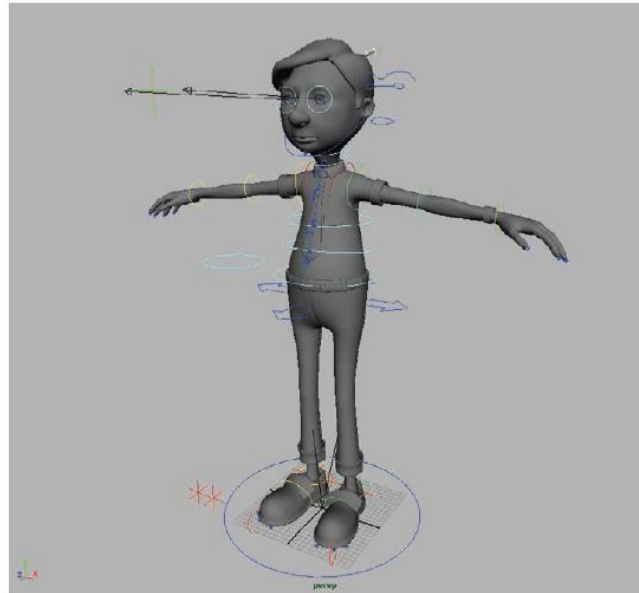


*Figure 7: Final model before textures (left) and after applying textures (right) [3]*

### **Rigging/Setup**

In rigging, a control rig is placed in an object so that the animators can move the object. The rigger's job is to create a system of controls that help the animators to work as quickly

and efficiently as possible. Every object that moves in a 3D animation project will have some kind of control system to control it.



*Figure 8: Rigged character [3]*

This control system can control a simple parent/child hierarchy, joints, controllers, skinning, a muscle system, and a floating GUI (Graphical User Interface) to help the animator's selections and keyframing. This component is very technical and requires an artist with good problem-solving skills. Figure 8 shows a final character object with control rig. The lines around the object are used by the animators and help them to move the object.

### **Animation**

In animation, the motion of objects or characters is created. Character animators make the audience believe that what they witness on screen is real and alive. Character animators must understand the physical properties of the character and implement the same in the project. The types of animation are keyframed animation (animator sets each pose and keyframes them with respect to time), path animation (animator sets a curve as a path for an object), motion capture (animator captures the motion from real-world from an actor to a control rig), and procedural animation (a programmer designs a set of rules and the object moves according to those rules). The animation component is the most time-consuming component of the production stage.

## **Visual Effects (VFX)**

The 3D visual effects (VFX) artist animates the character's properties and the surrounding environment, for example, fur, hair, cloth, fire, water, and dust. Most of the visual effects are based on a dynamic physics engine inside a 3D software that has natural properties such as air, gravity, and drag to manage these effects. So, a basic knowledge of physics and math is an absolute must. A 3D VFX artist works on different types of effects to improve the scene they are in, not to dominate them.

## **Lighting and Rendering**

The lighting artists look at the colour designs from the preproduction stage and insert the lighting to create a mood for a scene or sequence. Lighting in a 3D animation application is similar to the real-world lighting of film or photography. 3D artists have access to multiple light types that mimic lights in the real world, for example, ambient light, spotlight, area light, and sunlight. After setting up all the lights, the artist will break the whole project into render chunks, which are separate parts of the rendering process. These individual passes are joined together again in the compositing component of the postproduction stage.

## **2.3 Postproduction**

Postproduction is the last stage in a 3D animation project. This stage makes a project look polished and professional. Making changes in postproduction is more cost-effective than wasting hours of rendering time to fix a shot. Postproduction can be a fast and straightforward process if the preproduction and production teams planned for issues that may arise in postproduction. The postproduction stage has the following components:

- Compositing
- 2D visual effects
- Colour correction
- Final output

## **Compositing**

In compositing, the artists layer all the created and filmed imagery to make a final output image. The layering can be a simple task with only a few layers to operate, or it can be a

complicated task with hundreds of layers matched together. The final imagery can be 3D generated images or 3D and 2D mixed images.

### **2D Visual Effects and Motion Graphics**

2D visual effects and motion graphics are usually combined in the compositing stage. Both the compositor and 2D visual effects artists can be the same. The artist will add effects like transitions between the scenes, adding title cards in the beginning and at the end of the project. The motion-graphics artist's job is to generate the graphical design elements of a shot if required. This type of work can be viewed in film title sequences.

### **Colour Correction**

Colour correction also known as colour timing or colour grading, ensures that all scene colours are uniform. All 3D animation projects are created on as single shots, and then made as sequences and finally as a whole project. Every time the camera changes a shot is produced, and there are thousands of shots in the final film. Each shot needs to be adjusted to achieve the look of the film.

### **Final Output**

The final output of 3D animation can be in different types: film, video, rapid prototyping, 3D stereoscopic film and print media. Each of these output types has various workflows and technical limitations. But the most common output type is video that can be played on the computer.



## CHAPTER 3: LITERATURE REVIEW

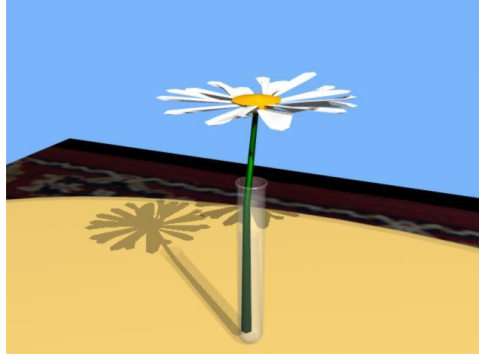
Prior work on 3D scene generation requires either pre-formatted input or uses annotated objects and scenes. The 3D models depicting real-world items must also be annotated with different characteristics so that priors can be used on their relationships with each other. Spatial relations play a crucial role in placing the objects in a scene that shows how the objects are connected. Prior research has presumed that either this world knowledge is supplied through manual annotation, or it has limited text-to-scene system discourse and production capacities to decrease the world knowledge needed.

Instead of having descriptions that describe the object positioning for individual scenes either by having manual annotations or giving input to the system in a different format, this thesis focuses on showing that natural language can be used to create 3D scenes by extracting the required information from text.

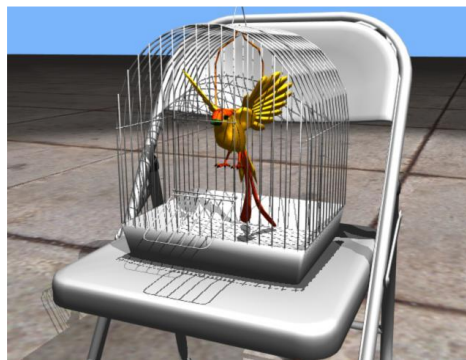
Based on the output type that is generated by a 3D scene generation system, research work in this field can be classified broadly as text-to-image, text-to-scene, text-to-video, text-to-animation and their inverse process as well. In our work, we mainly focus on text-to-scene and text-to-animation; after several centuries of advancement in computer hardware and 3D animation softwares, we think is very timely to investigate in these areas.

### 3.1 Text to Scene

Words eye [15] is a pioneer among all the systems that generate 3D scenes from textual descriptions. WordsEye is based on a pipeline that converts text into a 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 is publicly available online at <https://www.wordseye.com/> [16].



*Figure 9: The daisy is in the test tube [15]*



*Figure 10: The bird is in the birdcage. The birdcage is on the chair. [15]*

A model database of 2000 3D objects is used by the 2001 WordsEye system. Most importantly, WordsEye's authors recognized important key aspects of semantics and graphic representation attributes that are required for this task. WordsEye's main contribution is to provide an extensive collection of the most significant semantics of objects needed for text to generate the scene: sizes, orientation, parts (which objects have to be resized or have their orientation changed while placing them in scene), spatial relations (which objects are placed in scene for "on" or "under" relationships), skeletal poses, and functional tags. These attributes are linked to input text semantic depictions developed with entities, actions, attributes and relations.

WordsEye is developed in LISP [17](LISP stands for list processing). WordsEye extracts information from input text by tagging and parsing [18] it. Then it creates a dependency structure which is converted into a semantic representation. Predefined depiction rules are used to place the 3D objects in a scene with the specified colour attributes and spatial

relations from a set of low-level depicors which are converted from semantic representation.

PUT [1] system is a basic idea for all the text to 3D scene generation systems. Due to the limitations in natural language processing and graphics before the development of PUT, animators encountered great difficulties for using natural language for geometric description. PUT offers an interactive modelling environment and a programming interface that combines natural spatial relationships with direct manipulation to place rigid objects in 3D scenes.

PUT system uses God's-eye viewer and viewer-centric models as two manipulation metaphors. The God's-eye viewer model uses world-based coordinates and the viewer-centric model uses the viewer's coordinate system. In world based coordinates, the coordinates of all the objects are related to a world while in the viewer's coordinate system, which can also be referred to as the camera's point of view, the world based coordinates are transformed to viewers based on which are viewed by the user. Object placement parameters are obtained from dynamic, non-descriptive structures called image schemas. Image schemas are structures based on human interaction patterns with the physical world, such as body movements, object interactions, and spatial orientations. Spatial relationships and motion are the most fundamental image schemas. To explain the structure and application of spatial concepts, image schema and image-transformations are used. Simple schemas can be combined to form a complex schema by image- transformations. Thus, to structure a group of complex concepts, a relatively small number of image schemas are required.

PUT is a C++ parser [19] that uses these image schemas for spatial placement of the objects. The description of a region relative to the reference object is much of the detail in specifying the most basic spatial relationships. The reference object may be a pre-existing object or an arbitrary point in two or three dimensions. PUT system uses the following terminology: The reference object is called landmark (LM), and the object being placed is called trajector(TR). The spatial relationships relate TR to the LM. To specify the placement of TR, placement verbs such as "put" are used. Following are some object placement commands used in this system.

Put “box1” on “box2” below “box3”.

hang “box1” on “box2” at (0, 10).

put ‘box1” above “box2” and “box3” [1]

Li et al [20] presented a text to 3D scene generation system that uses XML [21] as an input to Autodesk Maya for 3D scene generation. This system consists of three modules: language engine, object database and graphics engine. In the language engine, the semantic information about the objects and spatial relations is extracted from the input text. Li et al. performed manual annotation of text descriptions to create an XML which can be used as a parser that can be easily integrated into other programs. Figure 11 is an example of an XML file for the input text.

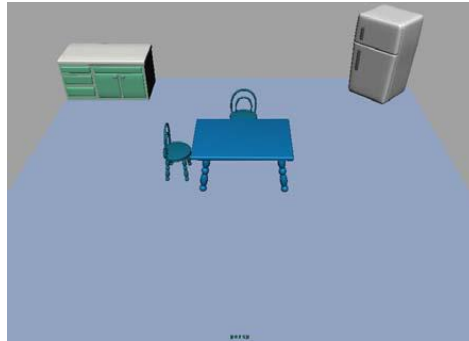
```
<?xml version="1.0" ?>
- <kitchen>
  <table attribute="middle" />
  - <chair01>
    <glorelpos relatedTo="table">directly-north-of</glorelpos>
    <reldis relatedTo="table">adjacent-to</reldis>
    <relori relatedTo="table">directly-facing</relori>
  </chair01>
  - <chair02>
    <glorelpos relatedTo="table">directly-west-of</glorelpos>
    <reldis relatedTo="table">adjacent-to</reldis>
    <relori relatedTo="table">directly-facing</relori>
  </chair02>
  - <fridge>
    <glorelpos relatedTo="table">northeast-of</glorelpos>
    <reldis relatedTo="table">near</reldis>
    <relori relatedTo="table">facing</relori>
  </fridge>
  - <cabinet>
    <glorelpos relatedTo="table">northwest-of</glorelpos>
    <reldis relatedTo="table">near</reldis>
    <absori relatedTo="air">facing-direct-south</absori>
  </cabinet>
</kitchen>
```

Figure 11: Example of XML file as input [20]

*“Table is in the middle of kitchen. One chair is to the north of the table and adjacent to and directly facing the table. Another chair is to the west of the table and adjacent to and directly facing the table. A fridge is to the northeast of the table and near and facing the table. A cabinet is to the northwest of the table and near the table, and directly facing south.”*

This XML document is given as input to the system to create a 3D scene. The entities represent the objects and the attributes indicate spatial relationships. Then the information in the XML document is transformed into a semantic web structure that shows how the objects are associated with each other with the spatial relations in a scene. For every object

in the semantic web, extract the spatial relation related to that object and place the object in scene according to it. The final kitchen scene from the input text is shown in Figure 12.



*Figure 12: Generated 3D scene [20]*

Lu et al. [22] proposed a New Framework for Automatic 3D scene construction from text description which is an extension of Li's [20] work where they used the bounding box to place the objects in a scene which helps in reducing collisions among objects in a relatively large scene.

Chang et al. [23] proposed the Interactive Learning of Spatial Knowledge for Text to 3D scene Generation system. It uses 3D models of indoor scenes that have text associated with them in the form of names and tags. They establish a probability of occurrence of a parent object when a given child object category is present in the input text. For example, by observing forks and plates on the table most of the time, we can determine that table is to be imported whenever there are plates and forks. The parent objects are segmented into planar surfaces to identify which surfaces on parent objects support child objects. The surfaces are categorized into up, down, left, right, front, on, back by the direction of their normal vector. Spatial relations between the objects are extracted in the form of *on*(A, B) or *left*(A, B) where A and B are objects, and on and left are spatial relations. The system uses Stanford CoreNLP pipeline to process the input text. By matching the words in the text against a list of known scene types from the scene dataset, the scene type is determined. To identify objects, the system looks for noun phrases and use the headword as the category. The Stanford coreference system is used to determine when the same object is being referred to. To identify object properties, adjectives and nouns are extracted in the noun phrase and also matched with dependency patterns to extract more attributes and

keywords. Considering the object category and attributes, and other words in the noun phrase mentioning the object, we identify a set of associated keywords to be used later for querying the 3D model database. To extract spatial relations between objects, dependency patterns are used.

In addition to the original input text given by the user to create a scene, a user can give textual commands to select and modify objects to filter the scene. For example, a user can give commands to “*remove the chair*” or “*put a pot on the table*”.

In continuation of the previous work, Chang et al. [24] proposed Learning Spatial Knowledge for Text to 3D Scene Generation. In this work, he made the system to learn object occurrence priors that help to determine the scene type. For example, the input is “*there is a knife on table*” then the system generates a kitchen scene with other related objects. Chang et al. modelled the relative position of objects based on their object categories and scene type.

Chang et al. [25] proposed a lexical grounding method to extract the weight of lexical features by training a classifier on a related grounding task to generate a scene. Then they combined this lexical grounding method with the previous method (rule-based model) for an improved scene generation system. Each noun phrase is extracted using the rule-based method; Chang et al. extracted features and computed the score of category. If the category is greater than 0.5, they select the best scoring category for representing the noun phrase. Otherwise, they use the head of the noun phrase for object category. They compare the results with previous models in five categories: *random*, *learned*, *rule*, *combo*, *human*. In the *random* condition, they select some random objects to place in a scene. The learned condition picks four most likely objects, and the rule condition uses a rule-based approach to generate a scene. The *combo* condition uses both lexical and rule-based methods to generate a scene. People create their own scene in *the human* condition. The comparison between the 3D scenes generated by different models can be seen in Figure 13.

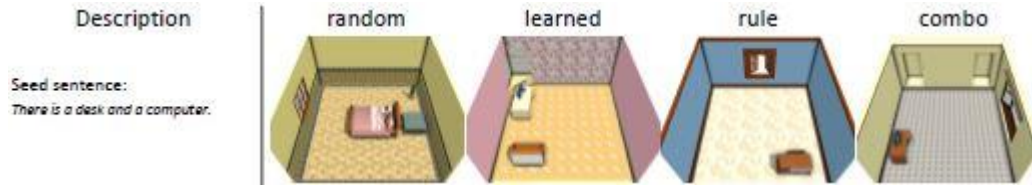


Figure 13: Scene generated using four different methods [25]

In 2017, Chang et al. proposed the SCEENSEER [26] method, which can modify or change the objects in generated scenes. Given the scene  $S$  and an operation  $O$ , we want to find a scene  $S'$  that is similar to  $S$  with the operation  $O$ . The proposed system supports *Select*, *Delete*, *Replace*, *Move*, *Scale* and *LookAt* operations: *Select* is used to select different objects, *Delete* is used to remove objects from the scene, *Replace* is used to replace one object with another, *Move* is used to change the location of the object, *Scale* is used to change the size of the objects and *LookAt* is used to make the camera focus on selected objects.

### 3.2 Text to animation

In contrast to static 3D scene generation, 3D animation generation is more complicated. Other than the problem of parsing natural language for semantic representation, there are additional challenges like the sequencing of events in an animation. Input text has to be analyzed to identify the objects and the actions associated with them. The most challenging part is planning the sequence of events so that the objects will not collide or change their paths unnecessarily.

Text to animation is notably more complex. Some of these structures attempt not merely to provide visual animation but also to generate of music and narration. Other than the problem of parsing natural language to semantic representation, there are extra challenges in storytelling, such as the selection of camera viewpoints, lighting, etc. Furthermore, planning for temporally extended sequences of actions introduces an even more complex setting for resolving ambiguities and constraints.

Many of the challenges of text to scene generation are also found in text to animation (also known as fiction to animation). They include: converting and grounding natural language textual content to meaningful representations, inferring implicit constraints, mapping the

words to graphics primitives and visualizing the primitives in the context of animation generation.

Another task that is made more challenging is the interpretation of verbs and actions. There are more implicit statements and unmentioned common-sense knowledge referring to the implications and constraints of everyday actions. For example, even producing a static image for “*Alex is brushing his teeth*” requires understanding that Alex need to be standing in front of a mirror and sink, with a toothbrush in his hand. Modern-day NLP [27] strategies are not yet adequate for computerized annotation of practical scripts describing stories to be animated.

On the graphics side, the automatic posing of characters and simulation of realistic human actions are much more complicated than the computerized design of static scenes. Consequently, posing and animation of characters in the entertainment industry is predominantly done through the manual effort of experienced 3D artists at a huge price. Automated systems are hardly ever entrusted with the project of depicting convincing human feelings in virtual characters without human help or verification. Chaining collective sequences of animations and planning for how they will be realized in a specific geometric environment is also challenging. Even a simple statement like “*The man grabs the book on the table*” is difficult if the man is sitting on a sofa on a different side of the room. It is essential to sketch a sequence of movements that makes the man stand up, stroll over to the book while realistically navigating around furniture, and then take hold of the book with a pose. Moreover, if we wanted to render some part of the broad range of human emotions, we would struggle to do so with modern graphics systems because a great deal of manual enhancing of character animations would be required.

Text to animation is the next step, involving generating static scenes. Adding animation to the static scenes makes it a complex and complete scene.

The CARSIM [28] system generates an animation of an actual car accident scene that summarizes the car accident reports written in French. It also implements linguistic analysis, i.e., extracts useful information from text, retrieves the objects from database that



are to be present in the scene and creates the 3D animation according to the scene description.

In CARSIM [28] system, the scene consists of three categories of objects: motionless objects (STATIC), moving objects (DYNAMIC), and collisions (ACCIDENT). STATIC and DYNAMIC objects describe the general environment in which the accident takes place. Each collision is represented by a relation between two objects that are either in DYNAMIC and/or STATIC state. Two parameters can be defined for a static object: one with the nature of the object and another with its location. The static objects that are involved in the accident are assigned an identity parameter(ID), while the ROAD objects have a KIND parameter which depends on the nature of the road. Possible KIND values for this system are: *crossroads, straight road, turn\_left, and turn\_right*. Trees, traffic lights, stop signs and pedestrian crossings are the other possible static objects. COORD parameter is used to provide their location. As trees and traffic lights can also participate in collisions, they also have an ID. Traffic lights have a COLOUR parameter to indicate the light colour (*red, orange, green or inactive*).

In the case of dynamic objects, they cannot be only by their nature and position. In addition to nature and position, their movement has to be defined. Every dynamic object is represented by a VEHICLE parameter, KIND parameter (car or truck) and a unique identifier ID. The movement of the dynamic object is defined by two parameters: INITIAL DIRECTION and EVENT. The INITIAL DIRECTION defines the direction of the object headed towards and the EVENTS parameter is an ordered list of atomic movements. CARSIM has the following events: *driving forward, stop, turn left, turn right, change lane left, change lane right, overtake*. The accident is described by using an ordered listing of collisions that take place in an accident simulation. A collision is described by means of giving the names of the two objects that participate in the collision and some additional attributes. These attributes are the collision location and the parts of the vehicles that are involved in the collision. The vehicle that collides is an *actor* and the vehicle that is hit is called *a victim*. A collision happens between an actor and a victim. The actor is a dynamic object and the victim can be either a static or a dynamic object. With all these parameters, the CARSIM system generates a car accident scene from the reports. For example, “*I was*

driving on a crossroads with a slow speed, approximately 40 km/h. Vehicle B arrived from my left, ignored the priority from the right and collided with my vehicle. On the first impact, my rear fender on the left side was hit and because of the slippery road, I lost control of my vehicle and hit the metallic protection of a tree, hence a second frontal collision” [28] is a converted text from French accident reports. This text is written as Formal Description and its respective output is shown in Figure 14.

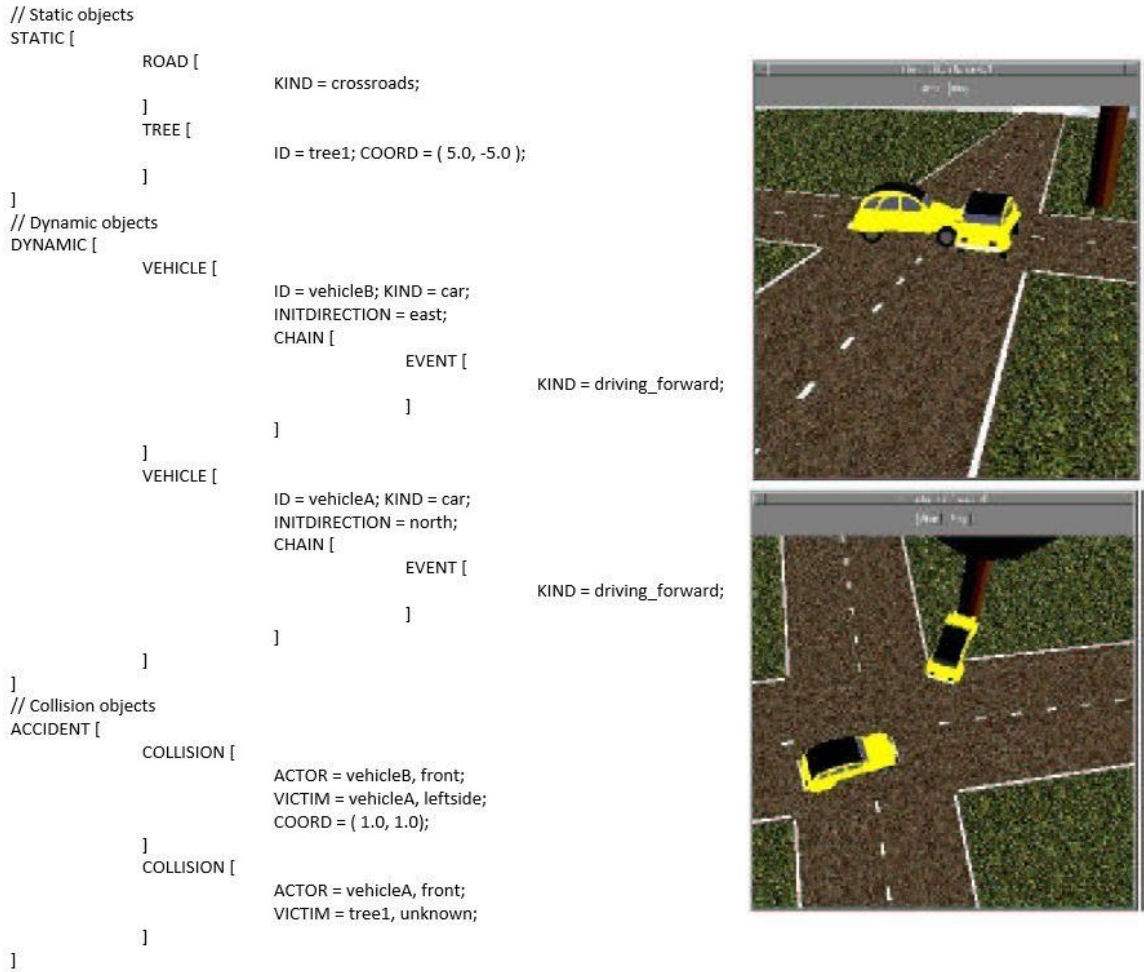


Figure 14: CARSIM system input(left) and output(right) [28]

Swan [29] proposed a rule-based system for generating an animation from a restricted form of Chinese natural language. This system uses five levels of representations, each with a corresponding language with well-defined syntax. The system takes input as a limited form of Chinese natural language called Moon Light, which is transformed into a frame-based semantics description language called Golden Forest. This language includes the

environment, characters, objects, and a sequence of actions extracted from the text. Using knowledge-based and pragmatic reasoning, the system constructs a qualitative script called Rainbow consisting of a sequence of temporally ordered scenes. Camera movements and lighting for each scene are also planned. The high-level qualitative script is then transformed into a quantitative script called Evergreen that specifies the series of static image frames to be generated and includes the concrete specifications of the objects, movements, camera, lighting, and actions. Finally, the quantitative script is converted into a low-level animation scripting language called SCRIPT that can be used to render the final animation. This system took a lot of manual effort to develop; over 30 graduate students contributed to the project over the course of nearly 10 years.

Glass [30] presented a system that transforms annotated fiction text into 3D animation. The annotated text contains the constraints like layout, appearance and motion associated with the objects in the scene.

Figure 15 is an example of annotated fiction text. To segment the input text into different scenes, annotations are used.

```
They had it on the top of a hill, in a sloping field that looked down into a sunny <setting>valley</setting>.
<avatar>Anne</avatar> didn't very much like a big brown <object>cow</object> who <transition
type='ARRIVE' subject='cow'>came</transition> up <relation type='near' subject='cow'
object='her'>close<relation> and stared at her, but it <transition type='DEPART'
subject='it'>went</transition> away when <avatar>Daddy</avatar> told it to.
```

*Figure 15: Annotated fiction text [30]*

Avatar and object annotations are used to select a list of objects that are to be present in each scene. Relation and transition annotations are later used to create a list of abstract constraints that summarize the layout of the scene. An entity descriptor is created for every avatar and object annotation that appears in a scene. Entity descriptors assign a geometric model sourced from a library of pre-built models to each avatar or object entity. Objects are fixed at an undetermined position, while avatars are assigned trajectories that provide movement to the objects. Co-references are resolved by maintaining the state that has the last male or female objects mentioned. Abstract constraints contain the object, their relations like *near*, *inside*, *no\_collide*, *behind*, *in\_front\_of*, *to\_left\_of* and *to\_right\_of* and

a time interval that specifies which constraint has to be executed during the animation. An example of abstract constraints is shown in Figure 16. Abstract constraints are then converted into mathematical constraints to generate a scene. Constraints with static objects are solved first, followed by dynamic objects.

CONSTRAINT 1: Subject: MAN Relation: OUTSIDE Object: ROOM Start-time: 0 End-time: 5	CONSTRAINT 2: Subject: MAN Relation: INSIDE Object: ROOM Start-time: 5 End-time: 30	CONSTRAINT 3: Subject: MAN Relation: NEAR Object: TABLE Start-time: 9 End-time: 14	CONSTRAINT 4: Subject: MAN Relation: NEAR Object: CHAIR Start-time: 14 End-time: 30
--	--	---	--

Figure 16: Example of abstract constraints [30]

Oshita [31] maintained a database of motion clips for various characters. The system extracts useful information and temporal constraints from input text and searches for an appropriate motion clip in the database. The motion clips are scheduled to make a whole animation according to temporal constraints. Firstly, syntax and semantic analysis are performed on an input text. Syntax analysis gives a tree structure with phrase tags and dependencies, while semantic analysis extracts the motion information from text. One query frame is generated for each motion in the text. Temporal constraints are created by the system that contains information about execution timing between motions. Motion scheduling determines the execution of each motion clip based on the temporal constraint. For each query frame, the system searches for the respective motion clip in the motion database. A motion timetable is generated that contains motions clips and their execution timings. This timetable is sent to motion synthesis which creates a final smooth animation with required objects. An example of Oshita's system and its output is shown in Figure 17.

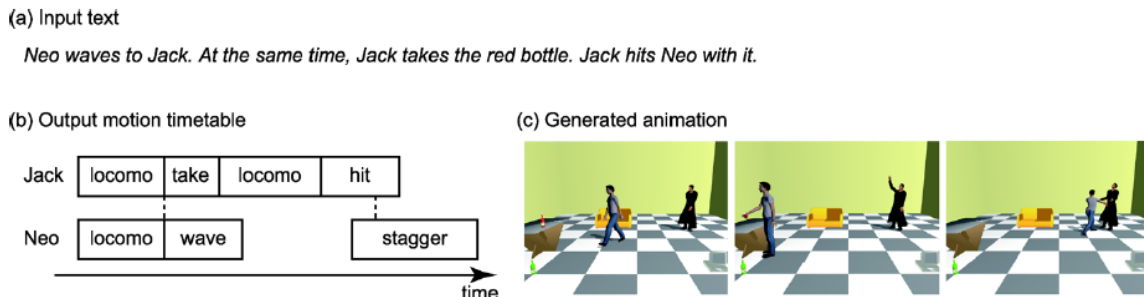


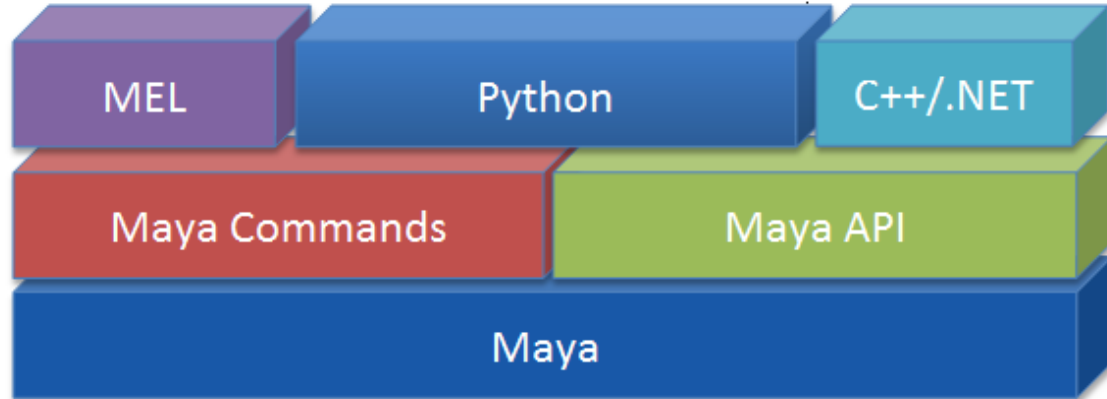
Figure 17: Example of Oshita's system [31]

## CHAPTER 4: AUTODESK MAYA

Autodesk Maya is a 3D animation software application which is used by most of the animation industries. Initially, Maya was developed by Alias Software systems in 1998 and later in 2005 it was acquired by Autodesk and renamed as Autodesk Maya. Since its initial release Maya has become widely used in film and gaming industries [32]. It is used to create 3D objects that depict real-world objects. It can also create visual effects that can be used in movies and games. Autodesk Maya deals with all the stages in the production pipeline from designing models to rendering them.

### 4.1 Maya Architecture

Maya is built on C++ API (Application Programming Interface) [33]. We can add new features or change the existing features by using Maya commands and Maya API interfaces. Maya commands interface can be developed by using MEL (Maya Embedded Language) or Python as a scripting language to write scripts. Maya API interface can be developed by using Python or C++ to write plugins for Maya.



*Figure 18: Maya Architecture [33]*

We are now going to go over all the programming languages and API's inside Maya. Maya has two API libraries. The first is the Commands library or cmds for short. It consists of bite-size commands that artists can use to make larger applications and tools. The other library inside Maya is called the OpenMaya API. This gives technical developers deep access to Maya's internal architecture, but it comes at the cost of complexity and it is not very suited for artists to pick up quickly.

MEL	Python	C++ and C#
Maya specific	Can be used outside Maya	Can be used outside Maya
Interactive	Can't be used interactively	Can't be used interactively
Used only for scripts	Used for scripts and plugins	Used only for plugins
	Slower than C++ and C#	Needs to be compiled
	Easy to learn	Much better performance

*Table 1: Pros and Cons of different programming languages in Maya [34]*

Maya has four programming languages. The first one is MEL [35]. It is a Maya specific language that is intended to be simple for artists to pick up, and it can only use a commands library. MEL is an interactive language, where we can type the commands and see the results in real-time inside the script editor in Maya.

This is opposite to Maya's other primary language C++ that was used to create Maya itself. C++ is a more complex language and it can only access the OpenMaya API. Additionally, C++ has to be compiled ahead of time, which means that it can only be distributed as plugins and can't be developed in real-time. Like C++ Maya has another language called C-Sharp. C# support in Maya is limited to Windows and it has the same caveats as C++. It has to be compiled and it can only access the OpenMaya API. However, C# is quite a bit easier to use than C++ and it is quite popular in the gaming community.

The fourth programming language in Maya is Python [36]. Python is in a unique position because it can access both the open Maya API and the commands library. It is the only language inside Maya that can do this. Unlike MEL, Python is used by programmers not using Maya as well. Python is quite popular in the programming community.

Google has chosen Python for its machine learning technologies, Dropbox uses it extensively, and scientists use it for scientific research. Python is popular for several reasons. It is an intuitive language while providing many advanced features that both C++ and C-Sharp have. This comes at the cost of performance. Python is much easier to use than C++ and C#, but it can be much slower. Usually, this is a tradeoff worth making



because Python code is quite easy to write and often does not need the speed that C++ or C# offer.

In this thesis, we used Python language for scripting in Maya to extend its functionalities. As not all the features that were provided by Maya can be accessed using Python, it became necessary to use MEL in between. MEL commands can also be used in Python scripts by calling `mel.eval()` function.

## **4.2 Python 2 vs 3**

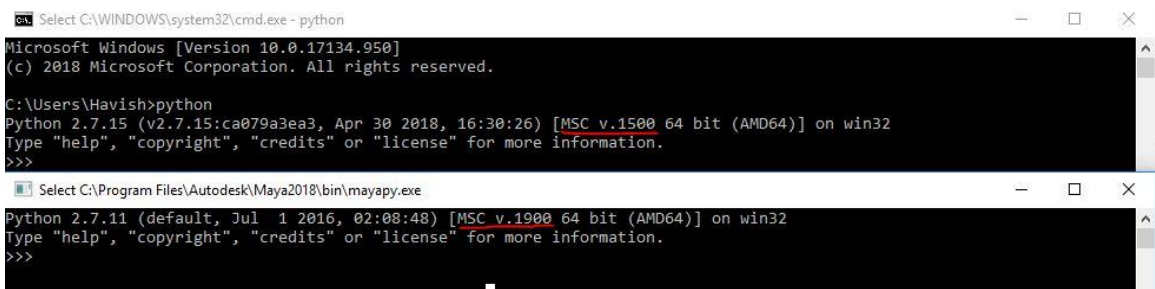
Python 2.7 is the final version in Python 2 series and Python 3.7 is the latest version in Python 3 series. Python 2 will have no additional active development except for some simple security fixes. Because there are many changes between the versions, they are not compatible with each other. Autodesk initially launched Python 2.4 in Maya 8.5. They have upgraded python versions to 2.7 until Maya 2014, and from that point, they are using the same version of Python.

The visual effects industry has decided to stick with Python 2.7 [37] until now because of significant investment in the Python 2 ecosystem which would be hard to port over to Python 3. The main reason for the visual effects industry to still use Python 2.7 is to minimize incompatibilities between different software packages. Therefore, to extend the functionality of Maya, we used Python 2.7 as our programming language.

## **4.3 Grounding Language**

As many animation softwares including Maya are still using Python 2.7, most of the advanced libraries are not supported [38]. The compilers inside these softwares are designed to perform the tasks that were meant for animation. The Python 2 packages for windows are compiled with Microsoft's Visual Studio 2008. However, Maya's Python interpreter is compiled in whatever version of Visual Studio Maya was compiled in. Figure 19 shows different versions of Visual Studio being used by Python in the system and in Maya. Note that MSC v.1500 and MSC v.1900 are the different compilers used for that interpreter. Different compilers compile incompatible code. That means that the dependencies of most of these libraries will crash when we try to run them through Maya. So to have special libraries that are not included with Maya we have to develop our own

libraries and build many of their dependencies from source using the appropriate version of Visual Studio which is a laborious and time-consuming process.



```
Select C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Havish>python
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

Select C:\Program Files\Autodesk\Maya2018\bin\mayapy.exe
Python 2.7.11 (default, Jul 1 2016, 02:08:48) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

*Figure 19: Different versions of Visual Studio installed inside Maya and system*

An extremely challenging job is to ground the natural language to detailed depictions of real-world entities. A visually situated framework can be used to address this challenge. In this framework, we improve system understanding by giving more information about the real world. We used visually situated framework for generating 3D scenes because there is a lack of common-sense language which connects language to everyday environments. For example, spatial relation "on" must be visualized depending on the context. Placing an object is different in both "on the wall" and "on the floor" situations.



## CHAPTER 5: METHODOLOGY

This chapter provides a review of basic concepts of the bounding box and spatial relations. Later in this chapter, we explain the framework of our system, its algorithm, scene generation and results.

### 5.1 Bounding Box

A bounding box is an invisible cuboid around every 3D object in 3D animation software that gives us the dimensions with respect to the X, Y and Z-axis. For an object in 3D space the bounding box gives  $X_{min}$ ,  $Y_{min}$ ,  $Z_{min}$ ,  $X_{max}$ ,  $Y_{max}$  and  $Z_{max}$  values, by which we can find the length of an object by using  $X_{min}$  and  $X_{max}$ , the height of an object by using  $Y_{min}$  and  $Y_{max}$ , the width of an object by using  $Z_{min}$  and  $Z_{max}$ . Knowing these dimensions helps us to prevent overlapping when we are placing an object. In Figure 20 there is an object table in the 3D space. To know the dimensions of the table we have to find `boundingBox('Table')` which gives the  $X_{min}$ ,  $Y_{min}$ ,  $Z_{min}$ ,  $X_{max}$ ,  $Y_{max}$  and  $Z_{max}$ . In general,  $(X_{max} - X_{min})$  gives length,  $(Y_{max} - Y_{min})$  gives height and  $(Z_{max} - Z_{min})$  gives width of an object where  $o$  stands for an object.

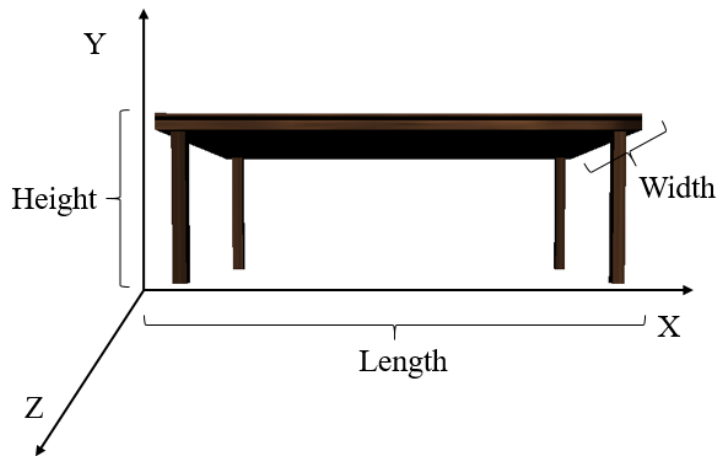


Figure 20: Table in 3D space with respect to X, Y, Z axis

Table  $(X_{max} - X_{min}) = \text{Length}$

Table  $(Y_{max} - Y_{min}) = \text{Height}$

Table  $(Z_{max} - Z_{min}) = \text{Width}$

## 5.2 Spatial Relations

Spatial relations are words that describe how objects are linked with each other in the real world. These play a crucial role in text to scene generation systems as they help to place objects in the 3D scene. Generally, spatial relationships are prepositions like on, in front of, under, etc. Table 2 shows different spatial relations that have been used in previously proposed systems.

3D Scene Generation Systems	Spatial Relations
Put [1]	in, on, at, above/below, front/back, left/right, next-to, left-of, right-of, against
WordsEye [15]	on, under, beyond, next to, in, in front of, right, left
Automatic 3D scene Generation Based on Maya [20]	front, direct-front, adjacent, near, north, west, northeast, northwest, direct-south, directly-north-of, directly-west-of, facing, directly-facing, facing-direct-south
Interactive Learning of Spatial Knowledge for Text to 3D Scene Generation [23]	left, right, above, below, front, back, on top of, next to, near, inside, outside

*Table 2: Spatial Relations supported by different systems*

We have categorized spatial relations into two categories: location and direction. Different locations around the object can be defined as: on, in front of, behind, inside, left, right, top and bottom. Directions include north, south, east, west, northeast, northwest, southeast and southwest. This categorization is for positioning the objects at a certain distance from other objects. When there is a location spatial relation between two objects, at least one face of the object is in contact with other object's face. When there is a direction spatial relation between two objects, one object is placed at a certain distance from the other object. By considering the spatial relations among the objects, their bounding box values are calculated and objects are repositioned in the scene.

Location spatial relations	Direction spatial relations
on, under, in front of, left, right, inside, behind, above, below, next	east, west, north, south, southeast, southwest, northeast, northwest

*Table 3: Spatial Relations supported by our system*

For example, in the input text, “*Left of the book there is a vase*”, left is a spatial relation between book and the vase. In order to place the vase to the left of the book, we consider the length of the objects and reposition the vase.

As another example, in the input text: “*Northeast of book there is a lamp*”, northeast is a spatial relation between the book and the lamp. Unlike the previous example, here we have to know both the length and width of the book to place the lamp diagonally across from it.

### **5.3 Framework**

Our proposed framework is shown in Figure 21. It is divided into two subtasks text processing and object positioning. In text processing, the system extracts useful information like object names and their spatial relations in the text using the Natural Language Tool Kit. In object positioning, the system fetches from the database all of the required objects identified in the previous step and imports them into the scene. After importing the objects, the system finds the bounding box values of every object in the scene to find the objects height, width and length. These dimensions and the spatial relationships that are extracted from text in the first step are used to place the object in scene. After object placement, if there is any motion associated with the object, it is added by adding key frames with respect to the time. Unlike the previous models, we have individual motion functions that add key frames to the object with respect to time indicates that an object is in motion.

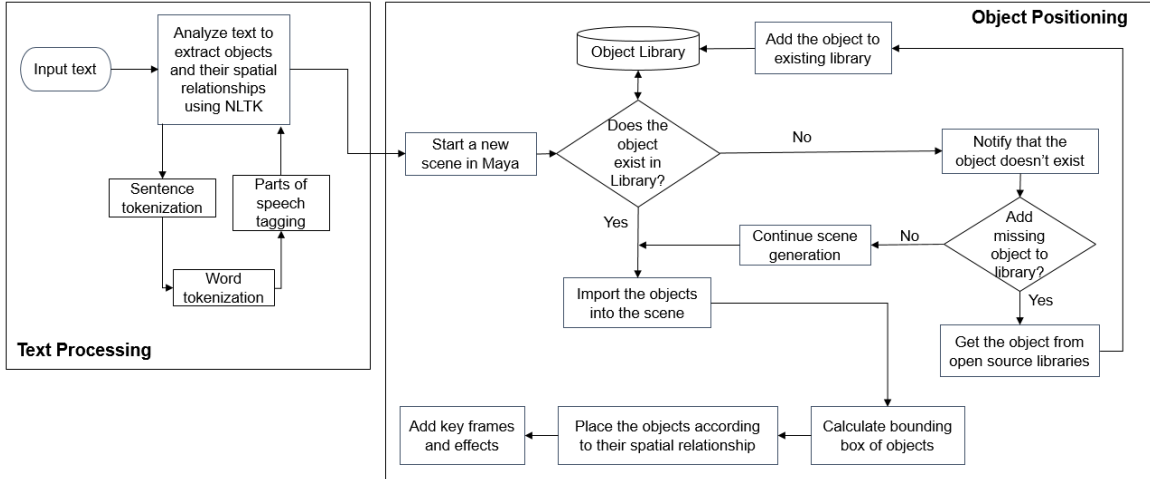


Figure 21: System framework

## 5.4 Algorithm

The stepwise algorithm of our proposed system for generating 3D scenes from text is discussed below.

---

### Algorithm 1: Dynamic scene creation from text

---

**Input:** Text of a scene description

**Output:** Generated 3D scene

- 1 Extract object names and spatial relations from the input text
  - 2 **if** *object exists in the database* **then**
  - 3     Import the objects into the scene
  - 4     Calculate the bounding box values of the objects to determine length, height and width of them.
  - 5     Find the 3D point  $p(x, y, z)$  at which the object has to be placed by calculating the distance that has to be there in between the objects  $(o_1, o_2)$  depending on the spatial relation  $R$ .
  - 6      $p(x, y, z) = \Delta(o_1, o_2, R)$
  - 7     Move the object to the calculated 3D point.
  - 8      $move(p, o_1)$
  - 9     Add motion and effects to the scene as described in the input text
  - 10 **else**
  - 11     Notify the user that object doesn't exist in the database
  - 12     **if** *the user wants to add the missing object to library* **then**
  - 13         Get the object from open source libraries and add it to the existing library
  - 14     **else**
  - 15         Continue with scene generation
  - 16 **End**
- 

Algorithm 1: Dynamic Scene Creation from Text

In step 5, calculating the 3D point to which the object has to move totally depends on the spatial relation and the location of the existing object. The detailed description for placing the objects is explained in the next section.

## 5.5 Scene generation

As discussed in the framework, our scene generation is divided into text processing and object positioning. First, we discuss the framework for text processing followed by the framework for object positioning.

### 5.5.1 Text Processing

In text processing, the system extracts the object names and spatial relations from input text. Usually, the object names are nouns and they are extracted by using Natural Language Tool Kit (NLTK). NLTK was created in 2001 as part of the computational linguistics course at the University of Pennsylvania by Edward Loper, Ewan Klein and Steven Bird [39]. NLTK is written in Python and distributed with an open-source license. It has dozens of algorithms that serve different functionalities and are used as the basis of many research projects. NLTK maintains a large and structured set of texts which is called a corpus. The algorithms are trained on these corpora (plural of corpus). In this thesis, we have used `nltk.tokenize` and `nltk.pos_tag` modules. Tokenize module is used to split the given text into sentences and sentences to words. Tag module is used to tag the words with respective parts of speech (POS). Below is an example of tokenization and POS tagging for the input text.

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

```
sentence_tokenized = nltk.sentence_tokenize (text)           //splitting the input text into sentences
```

```
⇒ ['Sphere is on cube',"cone is on sphere']
```

```
word_tokenized=nltk.word_tokenize(sentence_tokenized) //splitting each sentence into words
```

```
⇒ [['Sphere', 'is', 'on', 'cube'], ['cone', 'is', 'on', 'sphere']]
```

Parts\_of\_speech\_tagging=nlk.pos\_tag(word\_tokenized) //tagging parts of speech to every word

⇒ [(‘sphere’, ‘NN’), (‘is’, ‘VBN’), (‘on’, ‘IN’), (‘cube’, ‘NN’)]

⇒ [(‘cone’, ‘NN’), (‘is’, ‘VBN’), (‘on’, ‘IN’), (‘sphere’, ‘NN’)]

//where NN stands for noun, VBN stands for verb and  
IN stands for preposition

If any of the objects are not available, the user will be notified with a message saying that particular object is not found. The spatial relations are predefined. The spatial relations that were supported by our system are shown in Table 2. For example, "There is a vase on table" is the input text. Here, vase and table are the two objects and "on" is the spatial relationship between them. Figure 22, gives an example of a room scene that shows how objects are connected by the spatial relations with the input text "There is a chair in front of the table. On table there is a book. Left of the book there is vase. There is a flower inside the vase. Northeast of the book there is a lamp".

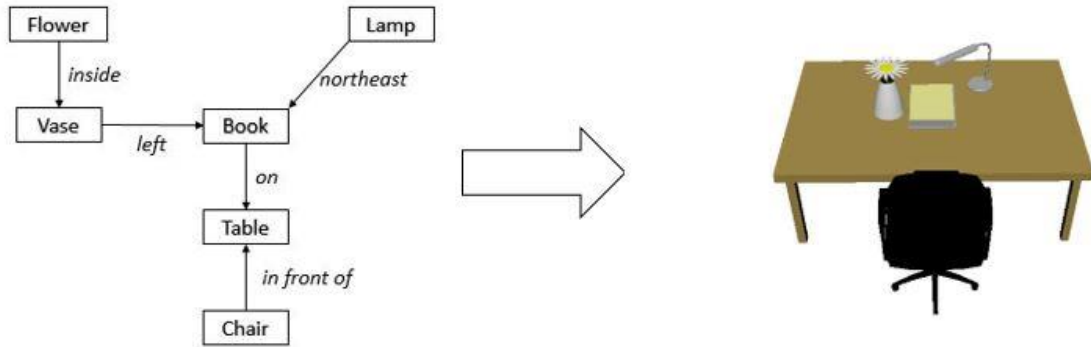


Figure 22: Representation of how objects are connected with the spatial relations and its respective final scene

If there is any motion or effect associated with an object, we extract the kind of motion or effect, and it is also applied to the object. We maintain a set of effects that can be included in the scene like snow, clouds, sky and fire. For example, "Building is on fire" is an input sentence; we made the system to create a scene where a building is burning instead of placing the building on fire. Here fire is an effect which applies to the object building. All this information that is extracted from the text is used for placing the objects in scene.

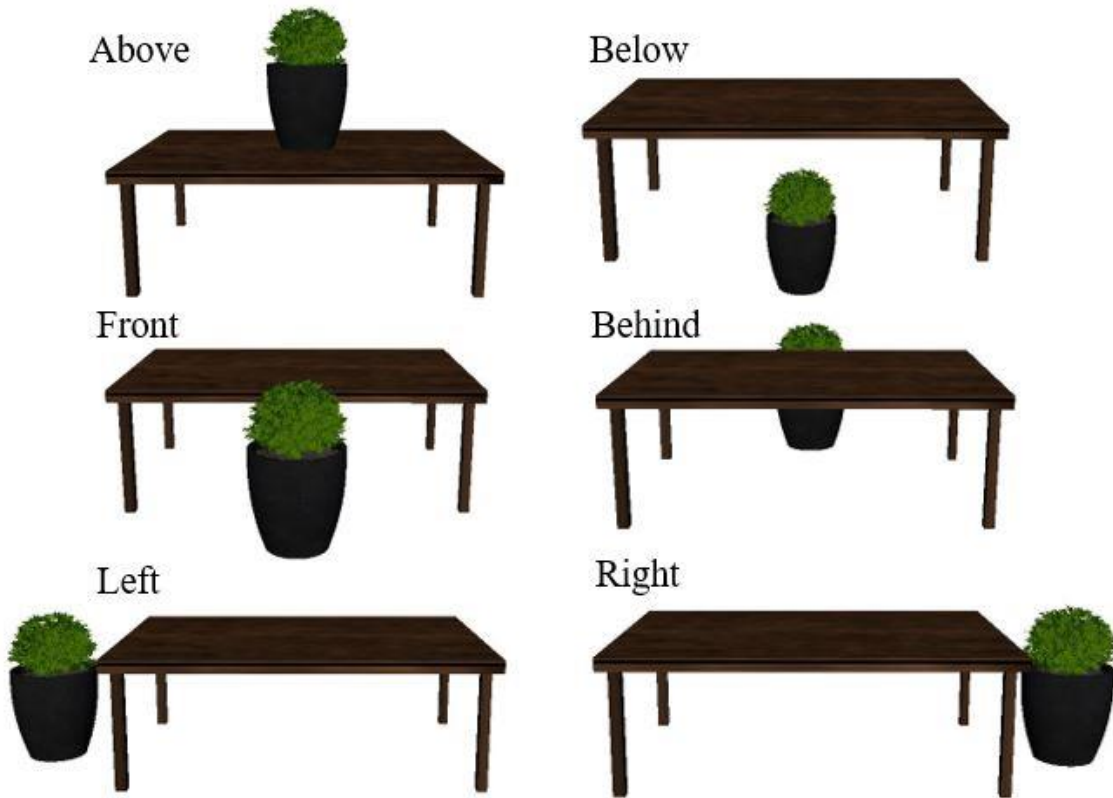
### 5.5.2 Object positioning

The next task after extracting object names and spatial relations is to generate a 3D scene. We maintain a database of individual objects in the local system. These objects can be modified by changing their size and colour attributes either before generating the final scene or after generating the final scene by using different tools available in Maya. We can also add new objects to this database or can replace the existing objects with new ones. Many artists in the animation industry are creating objects for their usage and making them available online; these objects can also be used in our system. Some of the 3D models that we have used are from TurboSquid, Highend3D, cgtrader and clara.io. All these companies provide thousands of objects for free that can be used by millions of users across different applications. These companies sell 3D objects under their terms and conditions where any kind of redistribution is prohibited.

As we have seen in Algorithm 1, object positioning is based on the object names and spatial relation extracted in the text processing, the system imports the 3D objects into the scene. Initially, all of the objects will be placed at (0, 0, 0) position in 3D space. After importing them, the system extracts the dimensions ( $X_{min}$ ,  $Y_{min}$ ,  $Z_{min}$ ,  $X_{max}$ ,  $Y_{max}$  and  $Z_{max}$ ) of the objects by using the bounding box function. Using these values, we find the height, width and depth of the objects. Depending on the spatial relationship between the objects we calculate either one or two dimensions among height, length and width, because not every spatial relation requires all three. For example, the spatial relation "on" requires the height of the objects to place them. The spatial relation "left" or "right" requires the length of the objects to place them and for the spatial relationship "northeast" requires both length and depth of the objects to place them in the scene.

To find the 3D point (X, Y, Z) at which the object has to be placed, we have to know the distance between the objects. To find the distance ( $\Delta$ ), we have to consider the objects' bounding box values and their spatial relation. For example, "*A plant is on the table*" is an input text for our system. Here plant and table are the objects and "on" is their spatial relation that is connecting. As we discussed, the system imports objects into the scene at (0, 0, 0) and it finds the boundingBox('plant') and boundingBox('table') for dimensions. Consider the table boundingBox values as (-1unit, 0unit, -0.5unit, 1unit, 1unit, 0.5unit), the

height is calculated by adding  $Y_{\min}$  and  $Y_{\max}$ . So, the height of the object is 1cm. This means that we have to place the plant 1cm above the table. In this scenario, the 3D point is (0, 1, 0) to which the plant has to be moved. Figure 23 shows the different object placement with the respect to spatial relations *left, right, above, below, front and behind*.



*Figure 23: Spatial relations with respect to a table*

We have seen an example of how we can place an object if it has to be moved only in one direction. What if it is a spatial relation like northeast or northwest or southeast or southwest where we have to consider two dimensions of an object to place it diagonally? Let's say "*the plant is northeast to the table*" is an input text. Now, we have to place the plant diagonally to the table. So, we have to consider both the object's length and width. As the direction is northeast we will add  $X_{\max}$  and  $Z_{\max}$  of both the table and plant.

Until now, we have seen examples that have single spatial relations. Let's say "There is a book on the table and a plant right of the book" is an input text. Here the book and table are connected with the spatial relation "on" and the plant and book are connected with



spatial relation "right". As explained before, the system finds the table height and places the book on the table. For placing the plant on the right of book, we have to consider the 3D point (X, Y, Z) of the book, as it is on the table and the plant also has to be on the table. So, we move the plant from the origin (0, 0, 0) to the location of book and then move it to the right of the book by finding the length of book and plant.

A particular spatial relation "on" can be used in two different ways such as "*a carpet is on the floor*" and "*a clock is on the wall*". In these two input statements, the spatial relation "on" has to place the objects in different positions by the system as it shouldn't place the clock above the wall. So, we have considered this as two different scenarios and implemented them in our system.

After object positioning the next step is to add animation and effects to the scene. In Autodesk Maya, there are different techniques to add motion to an object. We have used key frame animation which is one among them. In this technique, the object's location is saved with respect to the time. Instead of having the same movement for every object, we have different movements that can be assigned to any object in the scene so that the objects of similar type will have similar movement.

Consider the scene as depicted in Figure 26, where the man is walking and a car is moving. Usually, a car moves faster than a man, so the same movement can't be given to both the objects. We showed a difference in movement by giving more time for the man and less time for the car.

Finally, add effects to the scene. The system checks for any effects that are associated with the objects and adds them. For example, the input text is "*the building is on fire*". Here the fire is an effect that has to be added to the building, so the system adds the effect as shown in Figure 24 for the input text "*A building is on fire*". We consider this is also a unique condition where the system adds a fire effect to the object instead of placing the object on the fire.



*Figure 24: A Generated 3D scene with fire effect*

## 5.6 Results

### 5.6.1 Tools

<i>Item</i>	<i>Details</i>
OS	Windows
Languages	Python 2.7 & MEL
IDE	PyCharm
Text Processing	Natural language Tool Kit (NLTK) Python Library
Software	Autodesk Maya 2018
3D models	Created by myself and some from public repositories such as TurboSquid, clara.io and cgtrader

*Table 4: List of tools used for the implementation of our proposed system*

### 5.6.2 Generated 3D scenes

The proposed system has been tested on a small repository of 3D objects. Each of the generated 3D scenes is shown below with their given input text.

#### Input Text 1:

*“In front of the table there is a chair. On table there is a book. Left of book there is vase. There is a flower inside the vase. Northeast of the book there is a lamp”*

#### Generated 3D scene 1:



*Figure 25: Generated 3D scene for input text 1*

**Input Text 2:**

*“There is a car on road and car is moving. Left of the road there is footpath and a man walking on footpath. A building is left to footpath. The building is facing towards footpath. Building is on fire.”*

**Generated 3D scene 2:**



*Figure 26: Generated 3D scene for input text 2*

**Input Text 3:**

*“During sunrise the sky is full of clouds. Birds are flying above the mountains.”*

**Generated 3D scene 3:**

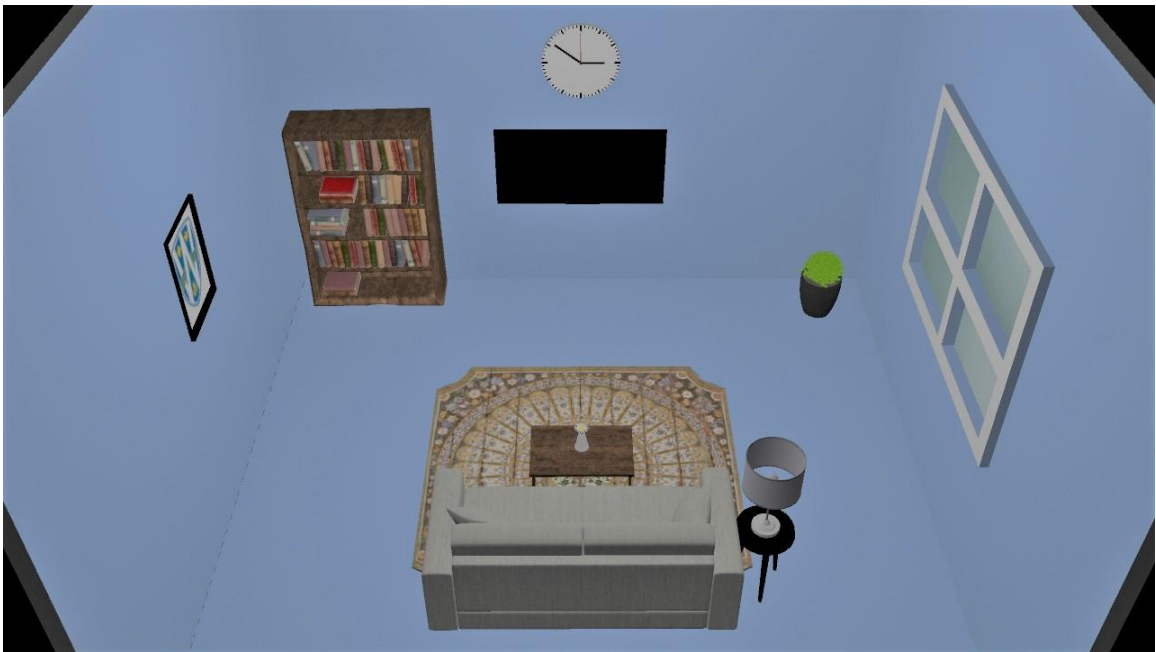


*Figure 27: Generated 3D scene for input text 3*

**Input Text 4:**

*“There is a table inside a room and a couch in front of table. Stool is right of the couch and a lamp on the stool. There is a vase on table and a flower inside 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 front\_wall and a TV below the clock. A photo frame is on left\_wall.”*

**Generated 3D scene 4:**



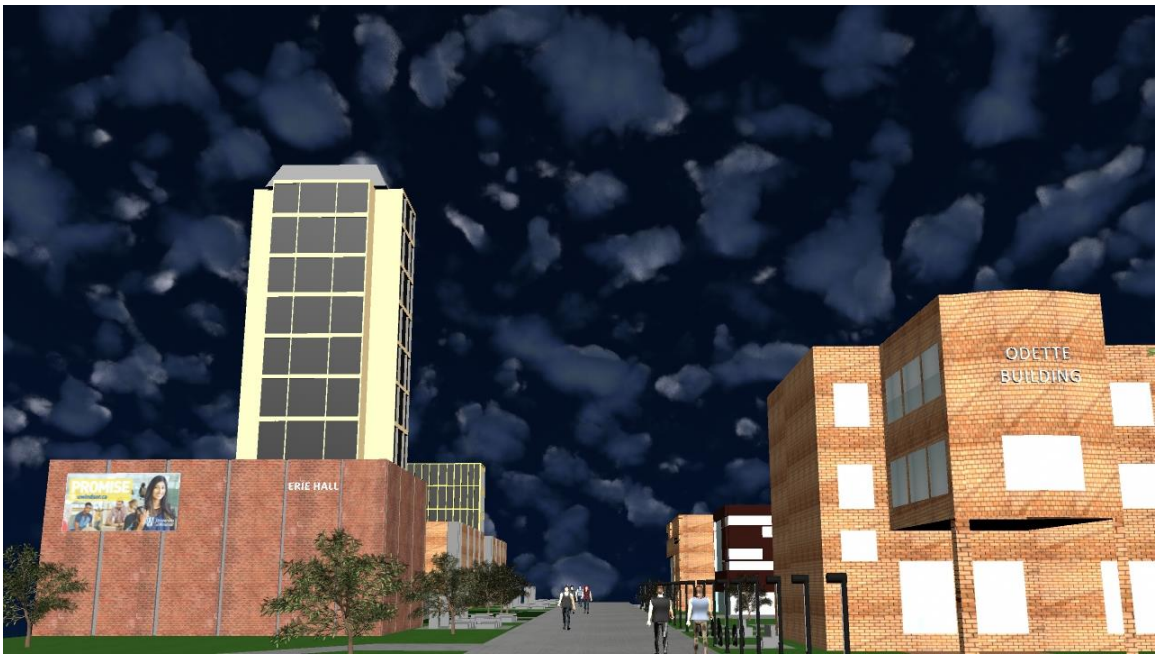
*Figure 28: Generated 3D scene for input text 4*



**Input Text 5:**

*“Odette building is on right of the Sunset avenue and Erie hall is on left of the Sunset Avenue. Next to Erie hall there is Lambton Tower and Chrysler Hall is north of Lambton Tower. North of Odette building there is Toldo building. Students are walking on the Sunset Avenue”*

**Generated 3D scene 5:**



*Figure 29: Generated 3D scene for input text 5*

## 5.7 Discussion

Since all of the text to scene generation systems use pre designed 3D objects we are not going to compare based on that feature but we will compare based on pre-formatted input, object positioning, animation and effects. Note that Pre-formatted input is a disadvantage to the system as the user has to develop a script from input text and that input is given to the system to generate 3D scene. In the table below, Y stands for *Yes*, indicating that the feature is included in the system whereas N stands for *No*, indicating that the feature is not included.

	Pre-formatted input	Object positioning	Animation	Effects
Put [1]	Y	Y	N	N
WordsEye [15]	N	Y	N	N
Automatic 3D scene generation based on Maya [20]	Y	Y	N	N
A New Framework for Automatic 3D Scene Construction [22]	Y	Y	N	N
Learning Spatial Knowledge for Text to 3D Scene Generation [24]	N	Y	N	N
SCENESEER [26]	N	Y	N	N
CARSIM [28]	Y	Y	Y	N
Automating the Creation of 3D Animation from Annotated Fiction Text [30]	Y	Y	Y	N
Generating Animation from Natural Language Texts and Semantic Analysis for Motion Search and Scheduling [31]	N	N	Y	N



Automatic Generation of Computer Animation [29]	Y	N	Y	N
Our system	N	Y	Y	Y

*Table 5: Comparison between different 3D scene generation systems*

For most of the 3D scene generation systems, the primary objective is object positioning with the help of spatial relations. As incorporating NLP into the animation industry is quite difficult and requires a lot of human effort, only a few 3D scene generation systems are using it. The systems that are not using NLP are writing the input text in the pre-formatted script designed for particular systems. None of the 3D scene generation systems have effects added to the objects. Effects that are supported by our system are snow, rain, sunrise, cloudy sky and adding fire to an individual object.

Li et al. [20] also developed a similar framework that generates a 3D scene in Maya. But the main difference is that the user has to know how to write a XML file. This XML file consists of objects' names as tags and attribute name-value pair for spatial relations. In order to generate a 3D scene with this approach, the normal text has to be converted to XML file by the user as shown in Figure 11 and then it is given as input. In our system, we can directly give text as input.

## **5.8 Conclusion**

We have proposed a framework for text to 3D scenes generation. We solved text to 3D scene generation problems by extracting the object names and spatial relations from the text and then placing the objects in the scenes according to their spatial relations. After placing the objects, the system also adds animation and effects to the objects to make the scene more realistic and complete. We have discussed bounding box, spatial relations, our framework, algorithm and scene generation system. As highlighted in Figure 30, we are able to complete major tasks in the production pipeline from layout to VFX by using our framework.

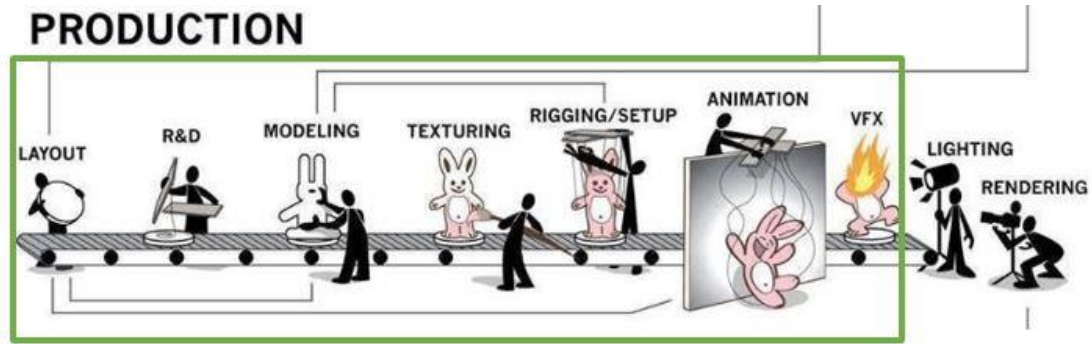


Figure 30: Tasks achieved in production stage by using our framework [3]

### Summary of Contributions

- Demonstrated how to handle input text with the help of NLTK by extracting object names and spatial relations.
- Explained how bounding box values are used to calculate length, height and width and used these values to place the objects in the scene by considering spatial relations.
- Demonstrated how animation and effects could be added to the objects.
- Considered the spatial relation "on" for different scenarios of placing the objects

This thesis is just a small step towards generating a 3D scene from text. There are many directions for future research which we discuss in the following section.

### 5.9 Future work

This thesis is just a small step towards generating a 3D scene from text. There are many directions for future research which we discuss in the following section.

The system can be improved in handling the input text. As indicated in Section 3.3, one can develop Natural Language Processing (NLP) libraries in Python within Maya. By developing these libraries, we can extract more information from the text and can perform semantic analysis on the complex input text to extract exact meaning from it. In its current form, it can process smaller sentences. There is also a need to extract information from relatively larger passages to build scene.

The library of objects we have created is rather small. There is a need to build a larger library and place it in public domain.

There is a need to add time series events or actions for the generated scenes. In time series events or actions, start or completion of an event or action depends on the progress of some other event or action. For example, in the text “As soon as I reached my house, it started snowing”, the event of snow fall starts when the person reaches his house.

We can also mention the distance at which the object has to be placed in a specific direction. For instance, *"There is a plant 1m right of the table"* is an input text. We can consider the distance 1m and place the plant on right side of the table.

We can also add audio in the scenes for respective effects like the burning sound when something is on fire, the sound of water drops when it's raining, etc.

During this implementation, we encountered a limitation for this system where one of the objects that we downloaded from the internet had bounding box values in exponential form. When we tried to place this object in the scene, it moved to an infinite position where we could not even find the object. To solve this problem, we added bonus tools plugin to Maya and used Bounding Box Scale module to give the Bounding Box values manually. Bounding Box Scale module is used for scaling objects to very specific unit size. This module scales the object in relative to the bounding box. Even after giving bounding box values manually, the object is moving to an infinite space while the values in Bounding Box Scale module are the same values that we gave.

## REFERENCES

- [1] S. R. Clay and J. Wilhelms, "Put: Language-based interactive manipulation of objects," *IEEE Computer Graphics and Applications*, vol. 16, pp. 31-39, 1996.
- [2] "Wikipedia," [Online]. Available: [https://en.wikipedia.org/wiki/Game\\_engine](https://en.wikipedia.org/wiki/Game_engine). [Accessed August 2019].
- [3] A. Beane, 3D animation essentials, John Wiley & Sons, 2012.
- [4] D. Errickson, T. J. U. Thompson, and B. W. J. Rankin, "The application of 3D visualization of osteological trauma for the courtroom: a critical review," *Journal of Forensic Radiology and Imaging*, vol. 2, no. 3, pp. 132-137, 2014.
- [5] "Autodesk," [Online]. Available: <https://www.autodesk.ca/en/solutions/cad-software>. [Accessed August 2019].
- [6] "Autodesk," [Online]. Available: <https://www.autodesk.ca/en/products/autocad/overview>. [Accessed August 2019].
- [7] "Autodesk," [Online]. Available: <https://www.autodesk.com/products/revit/overview>. [Accessed August 2019].
- [8] U. Buck, S. Naether, B. Rass, and C. Jackowski, "Accident or homicide—virtual crime scene reconstruction using 3D methods," *Forensic Science International*, vol. 225, no. 1-3, pp. 75-84, 2013.
- [9] "Nayoclinic," [Online]. Available: <https://www.mayoclinic.org/diseases-conditions/post-traumatic-stress-disorder/symptoms-causes/syc-20355967>. [Accessed August 2019].
- [10] "Wikipedia," [Online]. Available: [https://en.wikipedia.org/wiki/Finding\\_Nemo](https://en.wikipedia.org/wiki/Finding_Nemo). [Accessed August 2019].
- [11] "Wikipedia," [Online]. Available: [https://en.wikipedia.org/wiki/Monsters\\_vs.\\_Aliens](https://en.wikipedia.org/wiki/Monsters_vs._Aliens). [Accessed August 2019].
- [12] "Maya," [Online]. Available: <https://www.autodesk.com/education/free-software/maya>. [Accessed August 2019].

- [13] "Autodesk 3Ds Max," [Online]. Available: <https://www.autodesk.ca/en/products/3ds-max/overview>.
- [14] "Blender," [Online]. Available: <https://www.blender.org/>. [Accessed August 2019].
- [15] B. Coyne and R. Sproat, "WordsEye: an automatic text-to-scene conversion system," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001.
- [16] "Wordseye," [Online]. Available: <https://wordseye.com>. [Accessed August 2019].
- [17] "WhatIs," [Online]. Available: <https://whatis.techtarget.com/definition/LISP-list-processing>. [Accessed August 2019].
- [18] "Towards Data Science," [Online]. Available: <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72>.
- [19] "Techopedia," [Online]. Available: <https://www.techopedia.com/definition/3854/parser>.
- [20] C. Li, C. Yin, J. Lu and L. Ma, "Automatic 3D scene generation based on Maya," in *2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*, 2009.
- [21] "w3 info and knowledge," [Online]. Available: <https://www.w3.org/XML/>.
- [22] J. Lu, C. Li, C. Yin and L. Ma, "A new framework for automatic 3D scene construction from text description," in *2010 IEEE International Conference on Progress in Informatics and Computing*, 2010.
- [23] A. Chang, M. Savva and C. Manning, "Interactive learning of spatial knowledge for text to 3D scene generation," in *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 2014.
- [24] A. Chang, M. Savva and C. D. Manning, "Learning spatial knowledge for text to 3D scene generation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- [25] A. Chang, W. Monroe, M. Savva, C. Potts and C. D. Manning, "Text to 3d scene generation with rich lexical grounding," *arXiv preprint arXiv:1505.06289*, 2015.

- [26] A. X. Chang, M. Eric, M. Savva and C. D. Manning, "SceneSeer: 3D scene design with natural language," *arXiv preprint arXiv:1703.00050*, 2017.
- [27] "Tutorials Point," [Online]. Available: [https://www.tutorialspoint.com/natural\\_language\\_processing/index.htm](https://www.tutorialspoint.com/natural_language_processing/index.htm). [Accessed August 2019].
- [28] S. Dupuy, A. Egges, V. Legendre and P. Nugues, "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- Volume 13*, 2001.
- [29] R. Lu and S. Zhang, Automatic generation of computer animation: using AI for movie animation, Springer-Verlag, 2002.
- [30] K. Glass, and S. Bangay. "Automating the creation of 3D animation from annotated fiction text." In *Proceedings of the IADIS International Conference on Computer Graphics and Visualization*, vol. 3. 2008.
- [31] M. Oshita, "Generating animation from natural language texts and semantic analysis for motion search and scheduling," *The Visual Computer*, vol. 26, pp. 339-352, 2010.
- [32] "Edulearn," [Online]. Available: [https://www.edulearn.com/article/what\\_is\\_autodesk\\_maya.html#maya\\_training](https://www.edulearn.com/article/what_is_autodesk_maya.html#maya_training). [Accessed August 2019].
- [33] "Autodesk Maya Help," [Online]. Available: [http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=\\_\\_files\\_API\\_Introduction\\_htm](http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__files_API_Introduction_htm). [Accessed August 2019].
- [34] D. Govil, "Udemy," [Online]. Available: <https://www.udemy.com/python-for-maya/learn/lecture/6027296?start=105#overview>. [Accessed August 2019].
- [35] A. Maya, "Maya MEL commands," [Online]. Available: <https://help.autodesk.com/cloudhelp/2018/ENU/Maya-Tech-Docs/Commands/index.html>. [Accessed August 2019].
- [36] A. Maya, "Maya Python commands," [Online]. Available: [http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=\\_\\_CommandsPython\\_index\\_html](http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__CommandsPython_index_html). [Accessed August 2019].

- [37] "VFX Reference Platform," [Online]. Available: <https://vfxplatform.com>. [Accessed August 2019].
- [38] "3DeepLearner," [Online]. Available: <https://3deeplearner.com/binary-comp/>. [Accessed August 2019].
- [39] S. Bird, E. Klein and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*, O'Reilly Media, Inc, 2009.
- [40] X. Zeng, Q. H. Mehdi and N. E. Gough, "Shape of the story: Story visualization techniques," in *Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003.*, 2003.
- [41] P. Ye and T. Baldwin, "Towards Automatic Animated Storyboarding.," in *AAAI*, 2008.
- [42] L. M. Seversky and L. Yin, "Real-time automatic 3D scene generation from natural language voice and text descriptions," in *Proceedings of the 14th ACM International Conference on Multimedia*, 2006.

## VITA AUCTORIS

NAME: Havish Kadiyala

PLACE OF BIRTH: Vijayawada, India

YEAR OF BIRTH: 1995

EDUCATION: Bachelor of Technology in Computer Science,  
KL University, Vijayawada, India, 2016

Master of Science in Computer Science,  
University of Windsor, Windsor, ON, Canada,  
2019