SPEECH TO 3D SCENE GENERATION

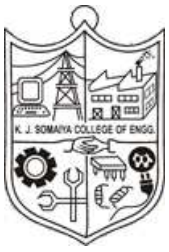
Project Synopsis

Submitted in partial fulfillment of requirements
For the degree of

BACHELOR OF INFORMATION TECHNOLOGY
BY

Manthan Sanjay Turakhia-1624013
Umang Nenshi Nandu-1624016
Prayesh Parag Shah-1624019
Siddharth Shashikant Sharma-1624020

Under the guidance of
Prof. Sagar Korde



Department of Information Technology
K.J.Somaiya College of Engineering, Mumbai-400077
(Autonomous College Affiliated to University of Mumbai)
Batch 2016-2019

Approval Sheet

Project synopsis entitled

Speech to 3D Scene Generation

Submitted By:

Manthan Sanjay Turakhia-1624013
Umang Nenshi Nandu-1624016
Prayesh Parag Shah-1624019
Siddharth Shashikant Sharma-1624020

In Partial fulfilment of the degree of B.Tech. in Information Technology is approved.

_____-
Guide

_____-
Examiner

_____-
Head of Department

_____-
Principal

Date :

Abstract

3D scenes and graphics are widely used in the creative industry. However, the entire task of imagination and then depicting the same as 3D graphics is done manually today, which consumes a lot of time, not to mention the inability to depict the scene precisely as imagined. We aim to reduce human efforts for the same by generating 3D scenes described by the user with precision, and near real-time generation. On the other hand, some industries currently lack the use of appropriate technology to make their tasks easier and more captivating, such as the education industry. We intend to replace the existing methods of teaching and learning by using speech to 3D scene generation to depict exactly what the professor is trying to explain.

Keywords: *Speech to Scene, 3D Warehouse, Linguistic Analysis, Spatial Relationship, Natural language processing.*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

This project plans to take working and understanding methods of various industries up a notch. This is to be achieved with the help of converting all desired speech to 3D scenes almost as-is. The goal is to improvise working, discussion and learning of education, corporate and creative industries by providing a better, and more attractive and interactive, platform to express thoughts and explain concepts, along with the ability to design various kinds of layouts and blueprints using only words. The end-game is for the users to be able to Literally paint the picture with words.

1.2 Motivation

The idea of this project was the result of a continuous observation of currently opted systems for expressing, understanding, learning knowledge and designing various designs. The current systems and methods were fading, resulting in monotony and ineffectiveness. Thus, the desire to make it all better was key for thinking and going forward with the project.

1.3 Scope

Ability to achieve near real-time 3D scene generation for provided speech, for various purposes and industries, with the help of a large collection of image and scene files. The users will also be able to view the generated scene from any angle (360 degree rotation). The project will achieve near-perfection status once it is able to learn common usual activities and usages in order to provide suggestions

and make predictions using Artificial Intelligence.

1.4 Salient contribution

Creating a dent in industries like education, corporate and creative, by drastically changing the way people express, understand and design their thoughts or work. The system is intended to be dynamic, generating 3D scenes on-the-go and as-is, with the ability to learn user habits for future benefits.

1.5 Organization of the Synopsis

The synopsis consists of introduction to the project. It contains the project management plan defining the tasks, descriptions, risks and schedule. It includes requirements for the project consisting of requirement specifications, UI, hardware and software requirements, and product attributes along with database requirements. It then contains the design description consisting of requirement traceability matrix, system architecture and UI. It also contains test plan and data for the project, along with test cases, approach and features to be, or not to be, tested.

Chapter 2

Literature Survey

2.1 Summary

We examine the task of speech to 3D scene generation. There is a myriad of applications for this technology, mainly for creative and educational industries. Designers can use this technology to interpret and display their thoughts and imaginations. Students can be taught with a near real-time graphical depiction of the topic. Commercial meetings and conference sessions can make the most of this technology.

2.2 Survey

The following observations were found in the literature survey:

1. Text to scene (limited capabilities).
2. Limited size databases (no dynamic generation or manipulation).
3. Scenes generated are not intelligent and precise hence, cannot be used for real-world applications.
4. Language used is unnatural.

The following papers were referred and used to understand the current systems and their working, and to derive knowledge of how implementation can be proceeded forward:

Will Monroe, 3D Scene Retrieval From Text With Semantic Parsing
Learning: Learnt the concept of semantic parsing from the text.

Wordseye:An Automatic Text-To-Scene Conversion System.
Learning: Learnt the linguistic analysis of text, and generation of the 3D scene itself. [1]

A Supervisory Hierarchical Control Approach For Text To 2D Scene Generation
Learning: Detecting changes and positioning of images and scenes.[3]

Real-Time Automatic 3D Scene Generation
Learning: Determining how to achieve real-time results and how to achieve the ability to detect input in users' natural language.[2]

Chapter 3

Software Project Management Plan

3.1 Introduction

3.1.1 Project Overview

The purpose of the software is to provide a better way for personnel form various industries like creative, corporate and education to present or impart knowledge in a better, more representative, and a more attractive way. As mentioned, the software is targeted for all kinds of professionals and students who are willing to make any kind of a presentation. The expected date of delivery is 19th of March, 2019.

3.1.2 Project Deliverable

Table 3.1: Project Deliverable

Delivery ID.	Deliverable/work products.	Delivery Date
D1.	SRS document which specifies the requirements for project.	30th Sept..
D2.	SPMP Document specifying over all planning and specifying the estimation.	30th Sept.
D2.	SDD Document specifying the designing of system.	5th Oct.
D2.	STD Document specifying the test cases and related information.	14th Oct.
D3.	UML diagrams.	31st Oct.
D4.	UI.	10th Nov.
D5.	Modules.	10th Jan.
D6.	Functional Prototype.	20th feb.
D7.	Application.	16th March.
D8.	Test Report.	25th March.

3.2 Project Organization

3.2.1 Software Process Model

Prototyping model
The chosen process model is Prototyping model. The Prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. This type of working is essential in our project because all the functional requirements need to be tested as a priority. Another important reason behind choosing this model is to make sure that at the end of the day the users get what they want. The software will be modified and updated until the end-game is achieved and the user is completely satisfied.

3.2.2 Roles and Responsibilities

Table 3.2: Roles and Responsibilities

Roles	Responsibilities
Team Leader (Umang Nandu)	Manage all the tasks and schedules the deadline
Project Manager (Siddharth Sharma)	Requirement gathering and coordination of various events.
Front-end Developer (Manthan Turakhia)	Development of user friendly user interface.
Back-end Developer (Prayesh Shah)	Development and linking of various back-end modules.
Tester (Umang Nandu)	Tests all the modules using software testing tools and techniques.

3.2.3 Tools and Techniques

1. Texworks to prepare project related documents.
2. IBM Rational Rose for Designing UML Diagrams
3. PyCharm for python programming.

3.3 Project Management Plan

3.3.1 Tasks

Table 3.3: Task breakup and associated deliverables

Tasks	Deliverables and Milestones	Resources needed	Dependencies and constraints
Gather Requirements .	SRS document which specifies the requirements for project.	Latex Editor	Users Approval
Confirmation of idea	SRS document specifies the functional and non-functional requirements.	Latex Editor	Stakeholders approval.
Planning	SPMP Document specifying over all planning and specifying the estimation.	Latex Editor.	Stakeholders and users involvement.
Content Audit.		Content Analysis Tool.	Evaluating content elements and information assets
Visual Design.	UI.		
Model Designing.	UML diagrams	IBM Rational Rose.	Approval from RTO.
Prototype Development.	Functional Prototype		Creating a basic functional prototype
Programming and Re-Engineering.	Modules.	Python IDE, Libraries, packages.	Gather end user feedback and alter if needed.
Linking.	Application.	Python IDE.	
Testing .	Test Report.	Unit Testing tools.	Constructed classes and various modules of the project.
Modification .			Approval of tester and end user.

3.3.2 Risk Table

Table 3.4: Risk Table

Risks	Category	Impact	Contingencies
Late Delivery	BU	2	Justification.
Computer Crash	TI	1	Accessing backups.
Technology will not Meet Expectations	TE	1	Taking feedback and modification.
Deviation from Software Engineering Standards	PI	3	Slight modifications if necessary.
Lack of Database Stability	TI	2	Making sure of a reliable database like Google 3D Warehouse.
Poor Comments in Code	TI	4	Separate manual for developers.
Users Disapproval	CRR	1	Using prototyping model.
Changes in Requirements	PS	2	Using prototyping model.
No internet connection	TI	2	

3.3.3 Time table

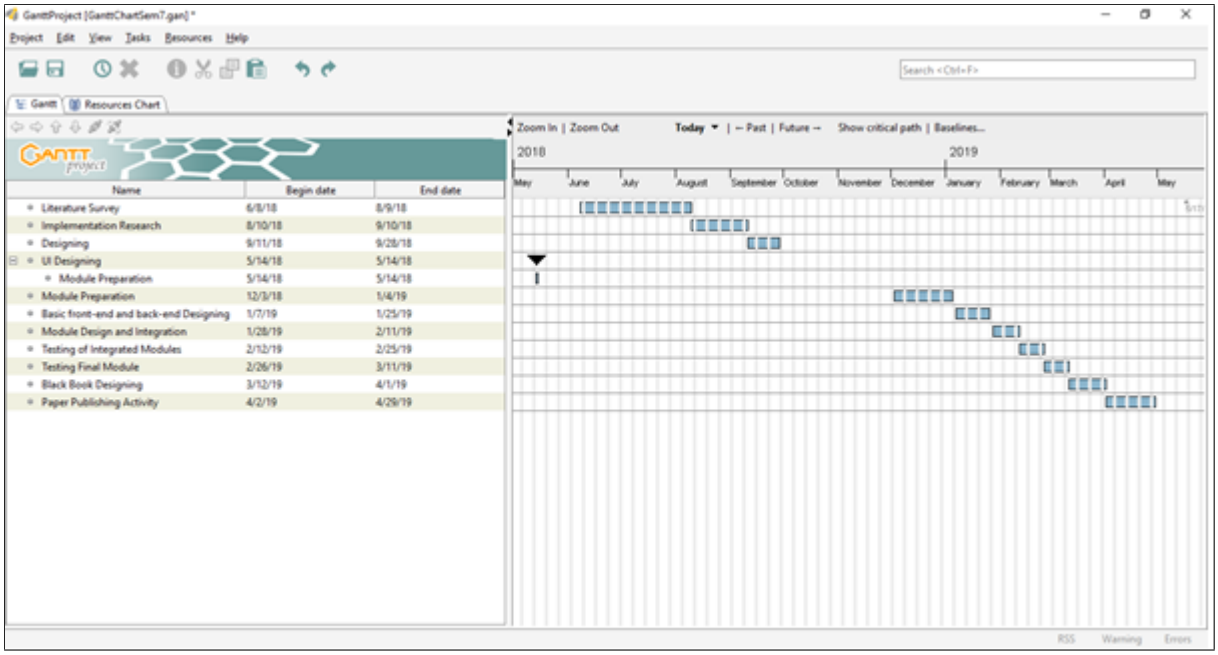


Figure 3.1: Project Time-line

Chapter 4

Software Requirements Specification

4.1 Introduction

4.1.1 Product Overview

"Speech to 3D Scene Generation" is a software that is developed to provide a near run-time digital/graphical output to the literal spoken words of the user. It goes through various stages before providing the final output. First, the speech is converted to text, then the text is passed to the interpretation protocol, and finally the interpreted text is used to render the output.
The best part of "Speech to 3D Scene Generation" is that it is meant to be used by any person who can speak. It is targeted to be used at various industries like education, creative, etc. as well as corporates. It will run on Windows Desktop Applications.
In general, the software will require APIs and coding platforms that will allow us to convert text to speech and then using speech to render the images.

4.2 Specific Requirements

4.2.1 External Interface Requirements

User Interfaces

The user interface requirements for title is are very general because it is a Desktop application. The PC at the user end should have only the basic screen layouts

with no requirements for latest OS. However, it may not be compatible for very earlier versions of Windows OS.
The user should be able to easily navigate to the part where it enables the speaker and the software should immediately start recording, converting and rendering. It is essential that it is simply a one-step process for the user and then it should all be a completely automatic process.

Hardware Interfaces

There isnt much hardware interfaces required since it is a completely software-oriented product. The only requirement is for it to work on any type of PC (Laptop, Computer) which match the basic OS and version requirements.

Software Interfaces

Softwares Required:

- Google Speech-to-Text API (Integrated Library)
- SpaCy Version 2.0.13
- 3D Warehouse/LFD Laboratory

4.2.2 Software Product Features

"Speech to 3D Scene Generation" will provide following features:-
Functional Requirements

1. Input Data requirements: :
    - Speech Input.
    - JSON as an input data to Database and Rendering.
2. Operational requirements
    - Conversion of speech to text.
    - POS tagging.
    - Parse tree generation.
    - Information gathering and rendering.

Non-functional Requirements

1. Performance: 75% conversion accuracy. Worst case 15s generation. Best case 3s.
2. Data Integrity: Data and modules to be kept abstract.
3. Usability: Smooth screen-to-screen movement.

4.2.3 Software System Attributes

Reliability

- Mean Time To Failure (MTTF) is Twenty Seconds.
- Expected optimal time for rendering and displaying is Seven Seconds.
- Speech-to-Text 75% accuracy.

Availability

- Failure at any point of the process will lead to complete termination and the user will have to start and perform the process all over again.

Security

- Since it is a Desktop application, the basic security measures taken by the user are sufficient with no additional requirements except for basic login credentials.
- Data/image/graph rendering is over the internet therefore simple internet security is more than enough.

Portability

- Entire software is mainly Python-oriented.
- No need of external compiler because of integrated environment.
- Most commonly used OS (Windows) is all that is required with no additional features.

Performance

- As mentioned, minimum 75% accuracy for Google speech-to-text API. Minimum latency for rendering.
- Users are expected to provide clear speech inputs, avoiding grammatical errors.

- Users are expected to be in a relatively quiet environment so as to ease the processing of the API.
- Data storage integrated using cloud therefore not much physical storage required.

4.2.4 Database Requirements

- No database required except for Google 3D Warehouse/LFD Laboratory.

Chapter 5

Software Design Description

5.1 Introduction

The design phase is aimed at the creative and adequate design of the project. The design phase focuses on the design and structure of the app, its relations with the main database as well as working of the app on a stand-alone basis. This phase also includes the information of various modules and diagrams which are used as figurative representation of the design and app working.

5.1.1 Design Overview

The architectural design is to be implemented in a manner which clearly conveys the flow of data and input/output streams. Each module is placed with respect to the actions performed by that module, the outcomes of the module, and the other modules affecting with those outcomes, so as to maintain the efficiency of the software with minimum time consumption. It can also be observed that some parts of the design have sub-modules as well such as geometric knowledge, linguistic knowledge, etc. which are going to be used by the parent module only, therefore in such cases it is made sure that these sub-modules do not interact and interrupt/interfere with the other parts of the design. During the software design phase, the implementation team will recommend how the system will be configured to support the industry needs.

5.1.2 Requirements Traceability Matrix

Table 5.1: Requirements Traceability Matrix

Functional requirements.	User.	User Accounts.	Server.
Login.	X	X	
Speech Input.	X	X	X
Database Manipulation.		X	
Data Storage.	X		X

5.2 System Architectural Design

5.2.1 Chosen System Architecture

Tier-1 Architecture

Figure 5.1 includes all the modules and interfaces involved in the project. The foremost step is to provide the speech input. This speech input will be received by the speech to text API, which will convert the provided speech to text. Next step is interpretation; this is where the converted text is being processed by SpaCy library used in Python environment. The term interpretation means that the text will be divided into multiple parts of speech identified by the library. As you can see, the library will make use of linguistic and word knowledge to precisely identify which word from the word knowledge (dictionary) fits into which category of the English language.
One the text is vividly classified, the database (3D Warehouse) comes into action. Note that the hierarchy created by SpaCy is the most important part. The database will use the hierarchy to identify the order in which the scene is to be generated and more importantly, the relation between the objects of the scene, along with the attributes of each object and the scene as a whole. It will require extensive geometric knowledge to place the objects exactly where required and also for mathematical purposes for forming a grid. Once all this is done, the scene will simply be rendered to the designed UI.

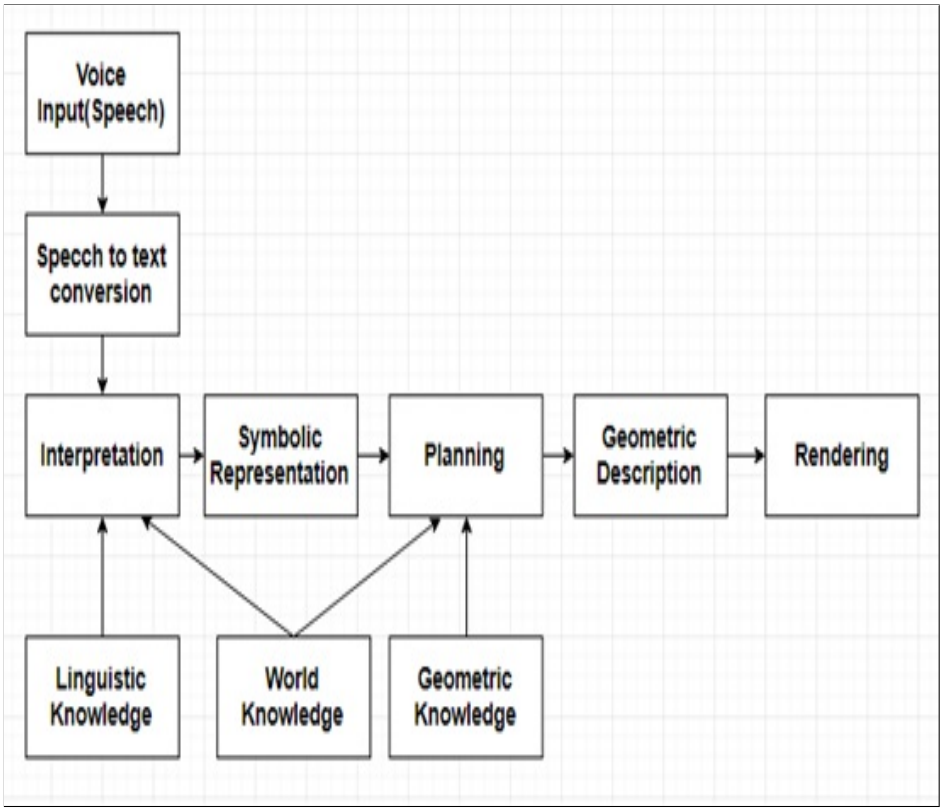


Figure 5.1: Architecture design

5.2.2 System Interface Description

The software is a desktop application and thus, will work on Windows OS. The user will simply need to install the software and it will be ready to use. All the libraries and database files will run in the background, potentially using cloud, therefore the user will only need to download and install the installer and main file.
The APIs and libraries used by the software are Speech to Text API, SpaCy library and 3D Warehouse database/dataset. All the components are back end products and therefore beyond users control and reach. The knowledge/information sets are also integrated with the APIs and libraries hence not concerning the user.

5.3 User Interface Design

5.3.1 Screen Images

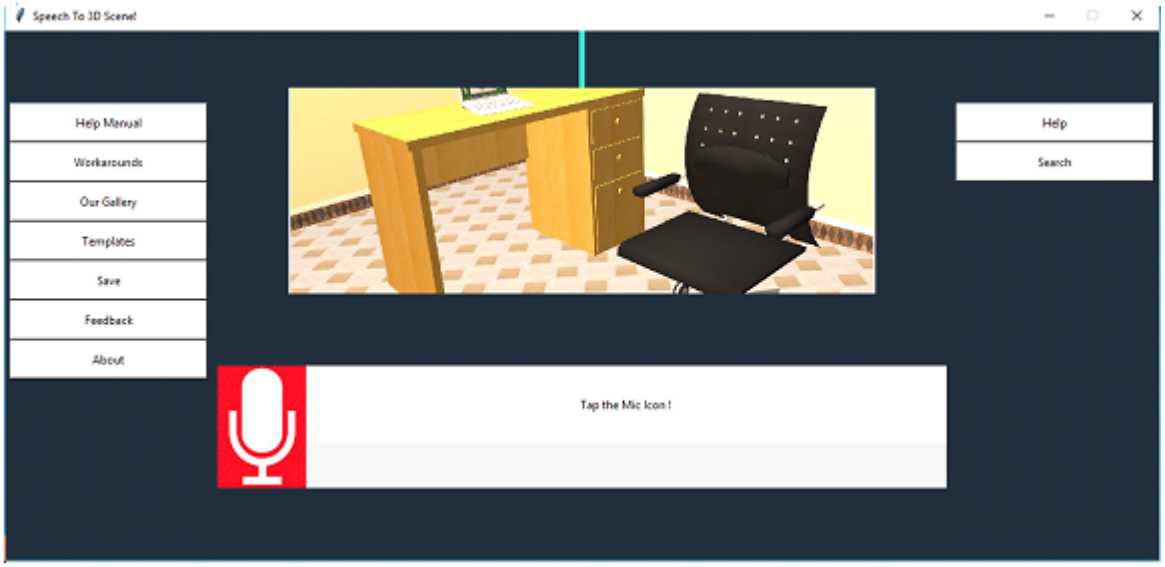


Figure 5.2: User Interface

5.4 Design Document

5.4.1 Level 0 DFD

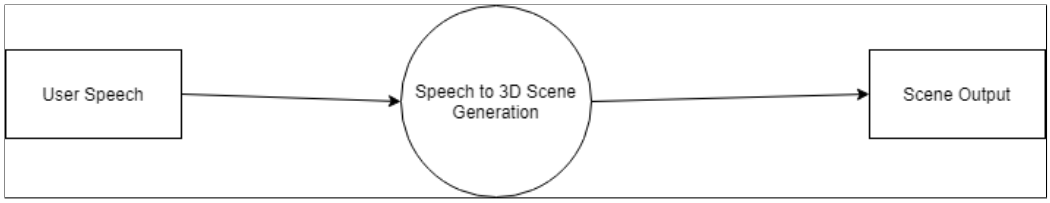


Figure 5.3: DFD Level 0

5.4.2 Level 1 DFD

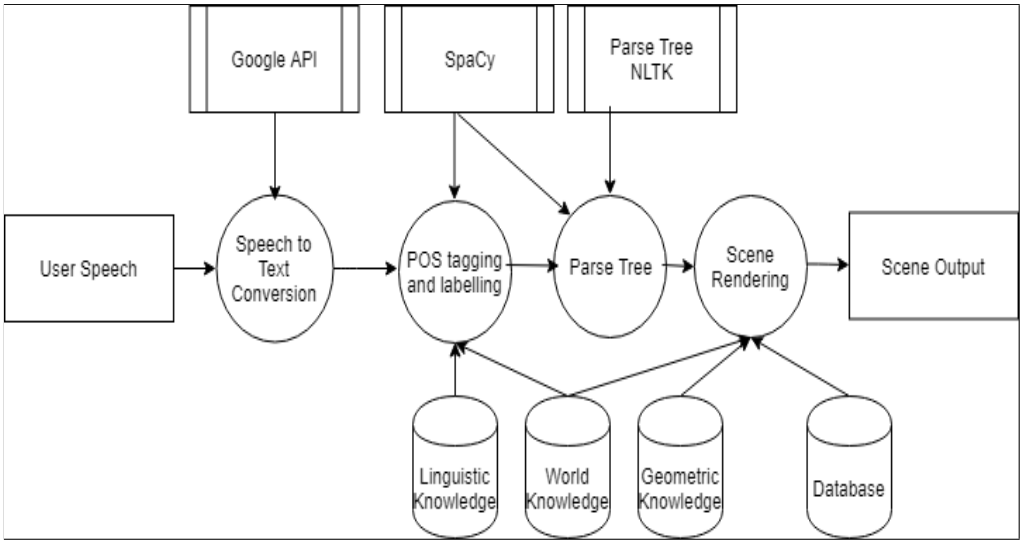


Figure 5.4: DFD Level 1

5.4.3 Use Case Diagram

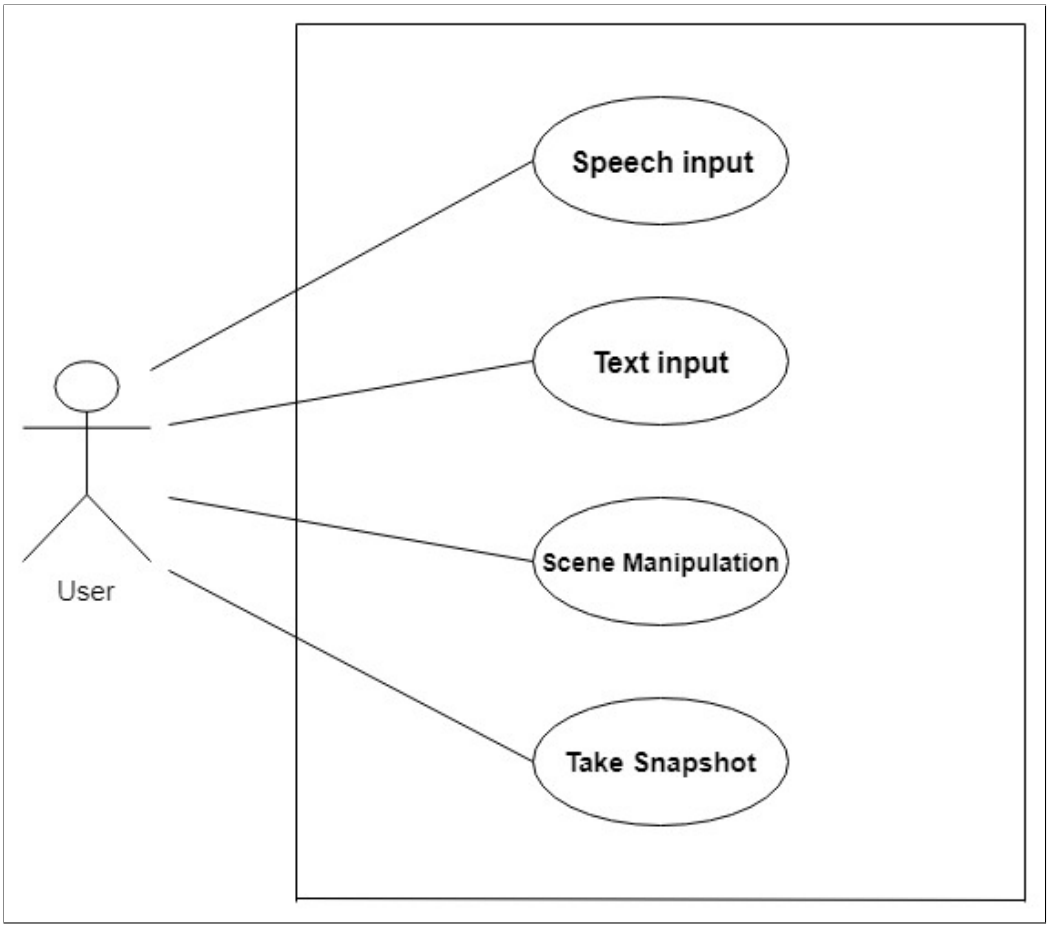


Figure 5.5: Use Case Diagram

5.4.4 Class Diagram

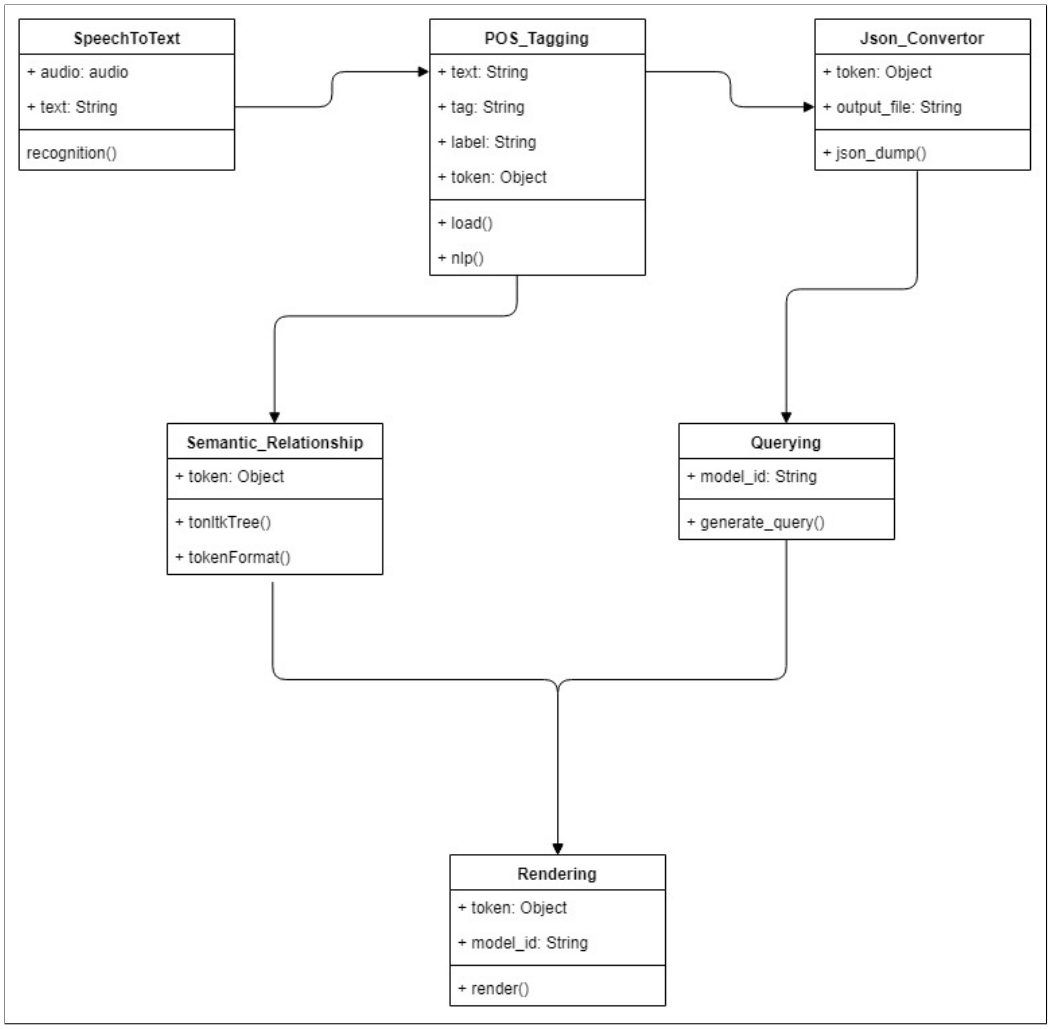


Figure 5.6: Class Diagram

5.4.5 Activity Diagram

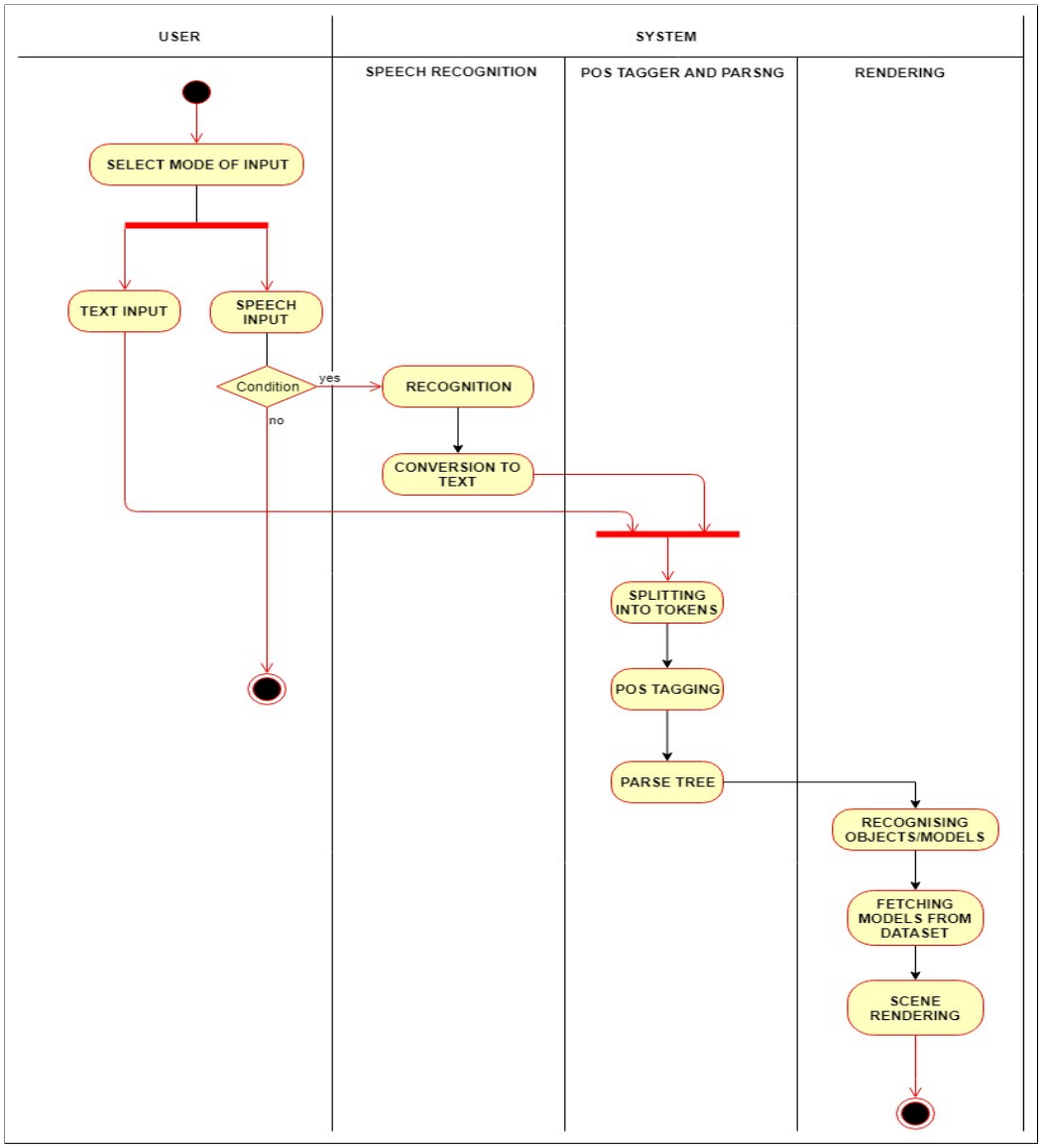


Figure 5.7: Activity Diagram

Chapter 6

Software Test Document

6.1 Introduction

6.1.1 System Overview

Speech to 3d Scene generator is a software application developed to provide a near real time 3D scene. It goes through various stages before providing the output. Speech input in converted into text, then the text is processed using natural language processing and broken down into low level parts of speech tags. A parse tree generating a semantic relationship is generated which will further be used to generate and fire queries dynamically to a 3d model database. The 3d models extracted from the model database will be rendered on a model viewer and positioned with respect to other models in that scene. Any queries fired further will manipulate the existing scene in near real time.

6.1.2 Test Approach

Table 6.1: Test Approach

TEST	DESCRIPTION.
Unit Testing	The purpose is to validate that each unit of the software performs as designed. Units like speech to text, parts of speech tagging, semantic relations and database handlers are tested.
Integration Testing	The purpose of this testing is to expose faults in the interaction between integrated units. Above mentioned modules are tested to remove faults.
Functional Testing	Includes testing of all database handlers. Input from the user is validated against various test cases.
Usability Testing	The application will be checked for user friendliness and comfort. Each user function is tested which includes test for navigation and buttons, content checking.

6.1.3 Features to be tested

1. **Speech to text Conversion**
2. **Scene Generation**
3. **Scene Manipulation**

6.1.4 Features not to be tested

1. **Parts of Speech Tagging.**
2. **Label are not to be tested.**

6.1.5 Testing Tools and Environment

Testing of the software application will require 15-25 days. Manual as well automated testing approaches will be applied.
1. AutoIT : AutoIT is a Stand Alone (doesnt require any configuration) and small footprint tool, that simulates mouse and keyboard clicks . It activates the binary files of the tested app using a Reflection.
The AutoIT comes with dedicated IDE, and is compatible with recordings and

coding in its own scripting language (very similar to BASIC syntax).

2. TestStack.White : White is a library for automation of desktop apps. It started as a small open source project and then became a part of TestStack which consists of a variety of open source code projects for automated and manual testing. White supports a variety of automation technologies: Silverlight, WPF, WinForms, Win32 and SWT in Java. Its possible to write White tests in any language supported by .NET.

3. Pywinauto : The PyWinAuto is a Python library that provides a collection of functions that make operations on Windows (controls and windows dialogs). The library presents a wide set of operations, is clear and user friendly.

6.2 Test Cases

Table 6.2: Test Cases

Test Case	Purpose	Input	Expected output
Speech to text conversion	Whether the input speech converted into the text is valid for the further processing or not. To check how accurate, the speech is converted is converted into text.	Voice Input	The text is valid if the text converted is same as the speech input given by the user. If it is then text is further processed else user can rerecord the input.
Tagging and Labelling	To check whether the parts of speech tagging and labelling of the text is done meaningfully or not.	Text Converted using Speech to text Recognition.	JSON or XML file which will contain proper parts of speech tagging and labelling of the text.
Rendered Models	To check whether the rendered models from data warehouse are perfectly suitable with the input provided in first stage.	No input from the user, the converted text is processed further.	Actual model of specific objects specified by the user are correctly rendered else final output will be incorrect, Models should not overlap.
Positions of the object (models)	To check whether the models rendered and displayed on the output screen are at proper coordinates as user wants.	No specific input, Text is processed	Objects are at proper position as mentioned by the user into the speech input.

References

[1] Bob Coyne and Richard Sproat. WordsEye: an automatic text-to-scene conversion system. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001.

[2] Lee M Seversky and Lijun Yin.Real-time automatic 3D scene generation from natural language voice and text descriptions. In Proceedings of the14th annual ACM international conference on Multimedia ,2006.

[3] R. Johansson, A. Berglund, M. Danielsson, "Artificial Intelligence" The Nineteenth International Joint Conference , pages 10731078, 2005.

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my guide Prof. Sagar Korde for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.
I also take this opportunity to express a deep sense of gratitude to Head of the department, Prof. Sujata Pathak for her cordial support, valuable information and guidance, which helped me in completing this task through various stages.
I am obliged to staff members of K. J. Somaiya College of Engineering, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of our assignment.

Date:

Manthan Turakhia
Umang Nandu
Prayesh Shah
Siddharth Sharma