Software Test Document
for
Speech to 3D Scene Generation

Prepared by
Manthan Turakhia - 1624013
Umang Nandu - 1624016
Prayesh Shah - 1624019
Siddharth Sharma - 1624020
Under the guidance of
Prof. Sagar D. Korde.

Contents

Chapter 1

Introduction

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.

1.2 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.

Chapter 2

TEST PLAN

2.1 Features to be tested

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

2.2 Features not to be tested

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

2.3 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.

Chapter 3

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.