Gesture Based URI Development
Voice Recognition Project
Submission Document

William Vida

1 Introduction

This is a submission document for the Voice Recognition Project as part of my final year module Gesture Based URI Development. This game allows the user to navigate through the menu using voice commands or using the mouse. The gameplay uses voice commands to control the character and the mouse to shoot. The grammar words and phrases are stored in XML files. On every scene, the valid words and phrases said by the user will be displayed on the screen. In this game, you take control of a character that moves automatically moves vertically and you must destroy three waves of enemies which also move automatically. This game was made using Unity 2020.1.10f1. The game is run using a resolution of 1920x1080. Any references for the code are in their respective files. For the build settings, the MainMenu scene is 0 while the GameScene is 1. Any known issues with the game are discussed in the Issues section.

2 Game Rules

2.1 Main Menu

The XML file that contains the grammar for the main menu is called MainMenu-Grammar.xml. The menu can be navigated using the mouse and by using any of the voice commands below. The main menu scene consists of four buttons, the play button, instructions button, a back button while you are in the instructions menu and a quit button. Each rule has a variation of what it should do. For example, the user can start a game by saying any of ”start”, ”start a game”, ”start the game” and more. Here are the grammar rules which are contained in the MainMenuGrammar.xml file:

```
<rule id="play">
    <one-of>
        <item>start</item>
        <item>start a game</item>
        <item>start the game</item>
        <item>click the start button</item>
        <item>play</item>
        <item>play a game</item>
        <item>play the game</item>
        <item>click play</item>
        <item>click the play button</item>
    </one-of>
</rule>

<rule id="instructions">
    <one-of>
        <item>instructions</item>
        <item>go to instructions</item>
        <item>go to the instructions menu</item>
        <item>click the instructions button</item>
    </one-of>
</rule>

 <rule id="back">
    <one-of>
        <item>back</item>
        <item>go back</item>
        <item>back to the main menu</item>
        <item>click the back button</item>
    </one-of>
</rule>

<rule id="quit">
    <one-of>
        <item>quit</item>
        <item>quit the game</item>
        <item>click the quit button</item>
        <item>click quit</item>
    </one-of>
</rule>
```

The words and phrases are recognised by a method called PhraseRecogniser() which is called by GR_OnPhraseRecognized() whenever a valid phrase is said by the user. PhraseRecogniser() uses a switch statement to determine what action must be done. This is common throughout the rest of the code for the other phrase recognisers in the game and pause menu. After each phrase is recognised, its respective action is delayed for one second using the Invoke() method. Here is the PhraseRecogniser() method from MainMenu.cs:

```
private void PhraseRecogniser()
{
    switch (phraseSpoken)
    {
        case "start":
```

```
        case "start a game":
        case "start the game":
        case "click the start button":
        case "play":
        case "play a game":
        case "play the game":
        case "click play":
        case "click the play button":
            Invoke("PlayGame", 1);
            break;
        case "instructions":
        case "go to instructions":
        case "go to the instructions menu":
        case "click the instructions button":
            Invoke("InstructionsButton", 1);
            break;
        case "back":
        case "go back":
        case "back to the main menu":
        case "click the back button":
            Invoke("BackButton", 1);
            break;
        case "quit":
        case "quit the game":
        case "click the quit button":
        case "click quit":
            Invoke("QuitButton", 1);
            break;
    }
}
```

2.2 Game

The phrases for the player's character and to access the pause are contained in the GameGrammar.xml file. The rules allow the player to access the pause menu, control the characters speed and choose the weapon. The pause menu can be accessed by voice or by clicking the escape key. The player can control the speed of the character or stop the character using their voice. There are four different speeds. The character can either be stopped, move at the default speed, move at the faster speed and move at the slower speed. There are four separate rules for each of them. The player has two different weapons which can both be accessed by voice commands. There are two different grammar rules to access the two weapons. Clicking the left mouse button will fire the weapon. Here are the grammar rules which are contained in the GameGrammar.xml file:

```
<rule id="pause">
    <one-of>
        <item>pause</item>
        <item>pause the game</item>
    </one-of>
</rule>

<rule id="move">
    <one-of>
        <item>move</item>
    </one-of>
</rule>

<rule id="speednormal">
    <one-of>
        <item>normal speed</item>
        <item>default speed</item>
    </one-of>
</rule>

<rule id="speedup">
    <one-of>
        <item>move faster</item>
        <item>go faster</item>
        <item>faster</item>
        <item>speed up</item>
    </one-of>
</rule>

<rule id="speeddown">
    <one-of>
        <item>move slower</item>
        <item>go slower</item>
        <item>slower</item>
        <item>speed down</item>
        <item>slow down</item>
    </one-of>
</rule>

<rule id="stop">
    <one-of>
        <item>stop</item>
        <item>stop moving</item>
    </one-of>
</rule>

<rule id="weaponone">
    <one-of>
        <item>weapon one</item>
        <item>main weapon</item>
        <item>fireball</item>
        <item>primary weapon</item>
    </one-of>
```

```
</rule>

<rule id="weapontwo">
    <one-of>
        <item>weapon two</item>
        <item>secondary weapon</item>
        <item>purple pearl</item>
    </one-of>
</rule>
```

The PlayerController.cs file is where the phrases are recognised. Like for the main menu, the phrases are recognised by a method called PhraseRecogniser() which is called by GR_OnPhraseRecognized() whenever a valid phrase is said by the user. PhraseRecogniser() uses a switch statement to determine what action must be done based on what was said. Here is the PhraseRecogniser() method from PlayerController.cs:

```
private void PhraseRecogniser()
{
    Weapon weapon = gameObject.GetComponent<Weapon>();

    switch (phraseSpoken)
    {
        case "weapon one":
        case "main weapon":
        case "fireball":
        case "primary weapon":
            weapon.selectedWeapon = 0;
            weapon.currentWeaponText.text = "Current Weapon: Fireball";
            break;
        case "weapon two":
        case "secondary weapon":
        case "purple pearl":
            weapon.selectedWeapon = 1;
            weapon.currentWeaponText.text = "Current Weapon: Purple
                Pearl";
            break;
        case "move":
            speed = defaultSpeed;
            break;
        case "normal speed":
        case "default speed":
            speed = defaultSpeed;
            break;
        case "move faster":
        case "go faster":
        case "faster":
        case "speed up":
            speed = upperSpeed;
            break;
        case "move slower":
        case "go slower":
        case "slower":
        case "speed down":
        case "slow down":
            speed = lowerSpeed;
            break;
        case "stop":
        case "stop moving":
            speed = 0;
            break;
        case "pause":
        case "pause the game":
            pauseMenu.Pause();
            break;
    }
}
```

2.3 Pause Menu

The pause menu is accessed from phrases from GameGrammar.xml but the navigation for the voice commands are contained in PauseMenuGrammar.xml. Three different rules are contained here, one for resuming the game, one to go to the main menu and another to quit the game.

```
<rule id="resume">
    <one-of>
        <item>resume</item>
        <item>resume the game</item>
    </one-of>
</rule>

<rule id="menu">
    <one-of>
        <item>menu</item>
        <item>to the menu</item>
        <item>go to the menu</item>
        <item>main menu</item>
        <item>to the main menu</item>
        <item>go to the main menu</item>
    </one-of>
</rule>

<rule id="quit">
    <one-of>
        <item>quit</item>
        <item>quit the game</item>
        <item>click the quit button</item>
        <item>click quit</item>
    </one-of>
</rule>
```

The PauseMenu.cs file is where the phrases are recognised. Just like for the previous grammar recognisers, the phrases are recognised by a method called PhraseRecogniser() which is called by GR_OnPhraseRecognized() whenever a valid phrase is said by the user. PhraseRecogniser() uses a switch statement to determine what action must be done based on what was said. Here is the PhraseRecogniser() method from PauseMenu.cs:

```
private void PhraseRecogniser()
{
    switch (phraseSpoken)
    {
        case "resume":
        case "resume the game":
            Resume();
            break;
        case "menu":
        case "to the menu":
        case "go to the menu":
        case "main menu":
        case "to the main menu":
        case "go to the main menu":
            LoadMenu();
            break;
        case "quit":
        case "quit the game":
        case "click the quit button":
        case "click quit":
            QuitGame();
            break;
    }
}
```

3 Issues

Unfortunately, there are some issues with the game that I know of.

- The pause menu grammar works while the game is playing which it should not.
- The game will start or quit if you use their voice commands while you are in the instructions menu.
- If the player pauses and clicks then the character will fire but it only happens for the first click and any subsequent ones will not.

4 Conclusion

In total there are 15 different grammar rules but some of them do the same thing and some other of them could be condensed together to create fewer grammar rules. I think that some of the rules are a bit unnecessary and should be scrapped but overall I think it was a good learning experience completing this project as I continued to improve my skills and knowledge of using Unity and C# by using new technology with Unity, voice recognition.