# Gesture Based UI Project Documentation

William Vida

Link to the GitHub repository: `https://github.com/WilliamVida/Gesture-Based-UI-Project`.

## Contents

# 1 Purpose of the Application

This game is a 2D endless platformer developed using Unity 2020.1.10f1. The game uses the keyboard and mouse and uses voice commands to accompany the game's mechanics. The language settings and keyboard should be set to US English and the microphone must be enabled. The game is run using a resolution of 1920x1080. In the build settings, the main menu scene is set to 0 and the game scene is set to 1. In this game, the player controls a character and must destroy enemies and avoid obstacles while also using generated platforms to collect coins to increase their score. The player loses if they collide with an enemy or an obstacle. The player's weapons are aimed at the mouse. If a valid phrase is recognised then it will be shown on screen but this is not necessarily a sign that the voice command did its intention.

# 2 Gestures Identified as Appropriate for This Application

The main menu consists of four buttons, to play the game, go to the instructions menu, quit the game and go back to the main menu from the instructions menu. Voice commands can be used to navigate through the menu along with the mouse. The phrases for the main menu voice commands are stored in MainMenuGrammar.xml. These voice commands are straight forward to use.

The pause menu consists of three buttons, to resume the game, go to the main menu and quit the game. The pause menu can be accessed by clicking the escape key on the keyboard or saying a variation of "pause" while playing. The pause menu buttons can be clicked by using a mouse or they can be used by using voice commands. The phrases for the pause menu are stored in PauseMenuGrammar.xml. The voice command to access the pause menu is contained in GameGrammar.xml which is for the game voice commands. These voice commands are straight forward to use.

The phrases for the main game are stored in GameGrammar.xml. The game allows the player to use voice commands to switch between any of the two weapons available. The option to reload can also be done using voice commands. These voice commands are straight forward to use. The two actions can also be done using the mouse and keyboard. The game contains power-ups that give temporary effects. They can be acquired by colliding the player with them or by saying a variation of the power-up while it is on screen. This was done to make it easier for the player as there may be unable to acquire them as there might be objects in the way.

# 3 Hardware Used in Creating the Application

I did not have any access to any special type of hardware such as a Kinect so I used voice commands from Unity using a microphone from my laptop. I had

ideas for games to develop using other hardware but I had to settle for using my laptop's microphone.

# 4 Architecture for the Solution

## 4.1 Main Menu Phrases

The main menu grammar phrases are recognised in MainMenu.cs which is attached to a canvas. If the method GR_OnPhraseRecognized() recognises a valid phrase then the method PhraseRecogniser() will be called. PhraseRecogniser() contains a switch statement which calls a method depending on the phrase said. The user can use the main menu buttons with a mouse or by using voice commands. The voice commands feel natural in this context and do not need much explanation. The code in PhraseRecogniser() is shown below.

```
private void PhraseRecogniser()
{
    switch (spokenPhrase)
    {
        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":
            PlayGame();
            break;
        case "instructions":
        case "go to instructions":
        case "go to the instructions menu":
        case "click the instructions button":
            InstructionsButton();
            break;
        case "back":
        case "go back":
        case "back to the main menu":
        case "click the back button":
            BackButton();
            break;
        case "quit":
        case "quit the game":
        case "click the quit button":
        case "click quit":
```

```
            QuitButton();
            break;
    }
}
```

The phrases for the main menu voice commands are stored in MainMenuGrammar.xml and the phrases are shown below.

```
<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>
```

## 4.2 Pause Menu Phrases

The pause menu grammar phrases are recognised in PauseMenu.cs which is attached to a canvas. If the method GR_OnPhraseRecognized() recognises a valid phrase then the method PhraseRecogniser() will be called. PhraseRecogniser() contains a switch statement which calls a method depending on the phrase said. The user can use the pause menu buttons with a mouse or by using voice commands. The voice commands feel natural in this context and do not need much explanation. To avoid an issue where the main game voice commands can be used while the game is paused an if statement is declared to check if the pause menu is active. The code in PhraseRecogniser() is shown below.

```
private void PhraseRecogniser()
{
    if (pauseMenuUI.activeInHierarchy ||
    ↪  gameOverUI.activeInHierarchy)
    {
        switch (spokenPhrase)
        {
            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;
        }
    }
}
```

The phrases for the pause menu voice commands are stored in PauseMenuGrammar.xml and the phrases are shown below.

```
<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>
```

### 4.3 Main Game Phrases

The main game grammar phrases are recognised in GrammarController.cs which is attached to an object called "Game Controller" in the game scene. If the method GR_OnPhraseRecognized() recognises a valid phrase then the method PhraseRecogniser() will be called. PhraseRecogniser() contains a switch statement which calls a method depending on the phrase said. The user can pause, change weapons and reload with either the keyboard, mouse or by using voice commands. Power-ups can be acquired by the player when the power-ups are visible in the main camera. If a valid phrase linked to the power-up is said then it attempts to find that object. If it is found and it is in the camera then the power-up effect is applied to the player. GrammarController.cs is attached to a game controller object. The code in PhraseRecogniser() is shown below.

```
private void PhraseRecogniser()
{
    if (!pauseMenuUI.activeInHierarchy)
    {
        GameObject weaponHolder = GameObject.Find("Weapon
        ↪  Holder");
```

```
WeaponSwitching weaponSwitching =
↪  weaponHolder.GetComponent<WeaponSwitching>();
Weapon weapon =
↪  weaponHolder.GetComponentInChildren<Weapon>();

switch (spokenPhrase)
{
    case "pause":
    case "pause the game":
        pauseMenu.Pause();
        break;
    case "weapon one":
    case "main weapon":
    case "primary weapon":
    case "assault rifle":
    case "rifle":
        weaponSwitching.selectedWeapon = 0;
        // To avoid an issue.
        weapon.canFire = true;
        weaponSwitching.SelectWeapon();
        break;
    case "weapon two":
    case "secondary weapon":
    case "pistol":
        weaponSwitching.selectedWeapon = 1;
        // To avoid an issue.
        weapon.canFire = true;
        weaponSwitching.SelectWeapon();
        break;
    case "reload":
    case "reload now":
    case "reload the weapon":
    case "reload the gun":
        StartCoroutine(weapon.Reload());
        break;
    case "more ammo":
    case "more ammo power up":
    case "more ammunition":
    case "more ammunition power up":
    case "extra ammo":
    case "extra ammunition":
    case "extra ammo power up":
    case "extra ammunition power up":
    case "get more ammo":
    case "get more ammunition":
    case "get the more ammo power up":
```

```
case "get the more ammunition power up":
case "get extra ammo":
case "get extra ammunition":
case "get the extra ammo power up":
case "get the extra ammunition power up":
    IncreasedAmmoPowerUp increasedAmmoPowerUp =
    ↪  FindObjectOfType<IncreasedAmmoPowerUp>();
    if (increasedAmmoPowerUp != null)
    {
        if (increasedAmmoPowerUp.isInCamera)
        {
            increasedAmmoPowerUp.transform.position =
            ↪  new
            ↪  Vector2(playerController.transform.position.x,
            ↪  playerController.transform.position.y);
        }
    }
    break;
case "fire rate":
case "more fire rate":
case "fire rate power up":
case "get the fire rate":
case "use the fire rate power up":
case "get the fire rate power up":
    IncreasedFireRatePowerUp increasedFireRatePowerUp
    ↪  =
    ↪  FindObjectOfType<IncreasedFireRatePowerUp>();
    if (increasedFireRatePowerUp != null)
    {
        if (increasedFireRatePowerUp.isInCamera)
        {

            ↪  increasedFireRatePowerUp.transform.position
            ↪  = new
            ↪  Vector2(playerController.transform.position.x,
            ↪  playerController.transform.position.y);
        }
    }
    break;
case "more speed":
case "more speed power up":
case "get more speed":
case "use the more speed power up":
case "get the more speed power up":
    IncreasedSpeedPowerUp increasedSpeedPowerUp =
    ↪  FindObjectOfType<IncreasedSpeedPowerUp>();
```

```
                if (increasedSpeedPowerUp != null)
                {
                    if (increasedSpeedPowerUp.isInCamera)
                    {
                        increasedSpeedPowerUp.transform.position
                        ↪  = new
                        ↪  Vector2(playerController.transform.position.x,
                        ↪  playerController.transform.position.y);
                    }
                }
                break;
            }
        }
    }
}h
```

The phrases for the main game voice commands are stored in GameGrammar.xml and the phrases are shown below.

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

<rule id="weaponone">
    <one-of>
        <item>weapon one</item>
        <item>main weapon</item>
        <item>assault rifle</item>
        <item>rifle</item>
    </one-of>
</rule>

<rule id="weapontwo">
    <one-of>
        <item>weapon two</item>
        <item>secondary weapon</item>
        <item>pistol</item>
    </one-of>
</rule>

<rule id="reload">
    <one-of>
        <item>reload</item>
        <item>reload now</item>
        <item>reload the weapon</item>
```

```
        <item>reload the gun</item>
    </one-of>
</rule>

<rule id="moreammopowerup">
    <one-of>
        <item>more ammo</item>
        <item>more ammo power up</item>
        <item>more ammunition</item>
        <item>more ammunition power up</item>
        <item>extra ammo</item>
        <item>extra ammunition</item>
        <item>extra ammo power up</item>
        <item>extra ammunition power up</item>
        <item>get more ammo</item>
        <item>get more ammunition</item>
        <item>get the more ammo power up</item>
        <item>get the more ammunition power up</item>
        <item>get extra ammo</item>
        <item>get extra ammunition</item>
        <item>get the extra ammo power up</item>
        <item>get the extra ammunition power up</item>
    </one-of>
</rule>

<rule id="fireratepowerup">
    <one-of>
        <item>fire rate</item>
        <item>more fire rate</item>
        <item>fire rate power up</item>
        <item>get the fire rate</item>
        <item>use the fire rate power up</item>
        <item>get the fire rate power up</item>
    </one-of>
</rule>

<rule id="morespeedpowerup">
    <one-of>
        <item>more speed</item>
        <item>more speed power up</item>
        <item>get more speed</item>
        <item>use the more speed power up</item>
        <item>get the more speed power up</item>
    </one-of>
</rule>
```

## 4.4 Player

- PlayerController.cs is attached to the player and controls the player's speed, speed increase multiplier, jumping abilities, animation and more.
- PlayerAimWeapon.cs points the player's weapons to the mouse position.
- Weapon.cs is attached to each weapon and controls the shooting, ammunition, reloading, fire rate, sounds and more.
- Bullet.cs is attached to a bullet prefab it sets the bullet's speed, damage and hit effect.
- WeaponSwitching.cs switches through the available weapons.

## 4.5 Game Controller

The scripts below are all attached to a game controller object.

- GameController.cs controls the starting point and checks if the player has died.
- GrammarController.cs controls the phrases for the game.
- Score.cs sets the score.

## 4.6 Spawners

- PlatformSpawner.cs spawns platforms and there is a chance if an obstacle spawns on it or a collectable coin to increase the player's score.
- EnemySpawner.cs spawns enemies at a certain rate.
- GroundObstacleSpawner.cs spawns ground obstacles.
- PowerUpSpawner.cs spawns the power-ups at a fixed interval.

## 4.7 Power-Ups

There are three different power-ups each with its own script.

- IncreasedAmmoPowerUp.cs increases the current weapon's maximum ammunition capacity until the next reload.
- IncreasedFireRatePowerUp.cs temporarily increases the current weapon's fire rate and reload speed.
- IncreasedSpeedPowerUp.cs temporarily increases the player's speed.

# 5 Issues

When changing weapons by voice and changing weapons using the keyboard or mouse, sometimes one or both weapons are unable to be fired. As the player goes faster, it becomes harder to acquire power-ups by voice as there is a delay by Unity when it recognises voice commands. This means that the power-up is already off-screen by the time the voice command is recognised by the game. Hence, the effects of the power-up are not used. The weapon aiming at the mouse is off by a few degrees at certain aiming angles.

# 6 Conclusions & Recommendations

Overall I enjoyed making this game. This is the second game that I have developed using voice commands and I did not meaningfully improve my knowledge of this feature. However, this game was also the first platformer that I have worked on and I gained valuable knowledge in this regard and I am happy that I finally made one albeit using voice commands. I would have liked to use other hardware rather than just the microphone and the Unity voice recogniser to challenge myself and develop a truly unique game. As this was a 2D platformer, I struggled to find voice commands that could be implemented in the game. This makes the voice commands rather generic. If I could do this project again, I would prefer to use another piece of hardware rather than using Unity's voice recogniser.

# 7 References

Every time I used code from another website, the link to the website is placed in its respective script. Two asset packages were used from the Unity Asset Store. The full list of references is below.

Free 2D Mega Pack (Asset) https://assetstore.unity.com/packages/2d/free-2d-mega-pack-177430.

Post Apocalypse Guns Demo (Asset) https://assetstore.unity.com/packages/audio/sound-fx/weapons/post-apocalypse-guns-demo-33515.

The power-up sprites were taken from https://callofduty.fandom.com/wiki/Perks.

The boosters for the player were adapted from https://www.youtube.com/watch?v=__y100uwVdM&ab_channel=Imphenzia.

The health bar was taken from https://www.youtube.com/watch?v=BLfNP4Sc_iA&ab_channel=Brackeys.

The button sounds were taken from https://www.youtube.com/watch?v=MjH5rsmYmQY&ab_channel=ElectronicBrai.

The main menu was taken from https://www.youtube.com/watch?v=zc8ac_qUXQY&ab_channel=Brackeys.

The pause menu was taken from https://www.youtube.com/watch?v=JivuXdrIHK0&ab_channel=Brackeys.

The background parallax effect was taken from https://www.youtube.com/watch?v=zit45k6CUMk&ab_channel=Dani.

The bullet class was taken from https://www.youtube.com/watch?v=wkKsl1Mfp5M&ab_channel=Brackeys.

The code to aim at the mouse position was taken from https://www.youtube.com/watch?v=fuGQFdhSPg4&ab_channel=CodeMonkey.

The weapon class was taken from https://www.youtube.com/watch?v=wkKsl1Mfp5M&ab_channel=Brackeys.

The weapon switching class was taken from https://www.youtube.com/watch?v=Dn_BUIVdAPg&ab_channel=Brackeys.

The power-up class was taken from https://www.youtube.com/watch?v=CLSiRf_OrBk&ab_channel=Brackeys.

The enemy and ground spawner was taken from https://www.youtube.com/watch?v=AI8XNNRpTTw&ab_channel=AlexanderZotov.

The platform spawner code was taken from https://www.youtube.com/watch?v=E7gmylDS1C4&ab_channel=PressStart.

Other platform mechanics were taken from https://www.youtube.com/playlist?list=PLiyfvmtjWC_XmdYfXm2i1AQ3lKrEPgc9-.