

Project for Gesture Based UI Development

G00358931 – Late Night Snack Game

Voice Gesture implementation

Purpose of the application

I wanted to create a game with a similar look and feel to the horror games of the early 2010's such as Slenderman and Amnesia however I wanted to try my hand at different functionality in games that I hadn't tried before, this one being a hunger system. The general goals of the above mentioned games is to find pieces of paper or clues before they get caught by a villain enemy AI. This game is similar in style as the player has to find gold coins to go to the take out shop and order food with the enemy being the hunger bar. The player must collect gold coins to buy food and survive for as long as possible. I Added some animations to the shop keeper who will greet the player as they enter the queue box collider which the player must be inside to order food.

I felt that if a user is to use a gesture other than the standard keyboard and mouse that it should compliment the games functionality rather than add extra stress or difficulty. It was important to identify whether I was using a phrase or gesture for the sake of having a gesture or having a gesture that makes sense in the games context , here is where I felt that ordering food with voice was appropriate and something the user could do with ease.

Gestures identified as appropriate for this application

I found that the most meaningful way to interact with the game was using what tools I had at my disposal appropriately. I decided that I should use my voice to order the food at the take out joint. It would make sense to use a voice gesture to emulate how we really would order food at a restaurant as apposed to a UI menu where you click the food you want, here we can ask for whatever food we want to eat so long as we have enough money and are in the queue. The phrases for ordering are stored in Grammar.xml. I felt that using voice controls for other aspects of the game would not really be appropriate like a command to eat or pickup an item as they are more easily more effective via mouse and keyboard inputs and having them done through

voice , although doable would ultimately make the game awkward and slow down the player.



Here the interacting with the food object makes more sense through a key down command rather than a voice phrase commanding the player to eat, especially as the games context is in a first person perspective. Talking to or at your own character doesn't make much sense compared to or at another in game character as most people use their voice to interact with other people in real world scenarios.

Hardware used in creating the application

I have a Logitech webcam however it is a very old model Logitech E2500 which released back in 2008. As a result it could not display a high enough resolution to work with unity in the Augmented Reality labs we did this semester and hence to implement a camera function was not an option unfortunately. I only had the option to use my headsets microphone which led me to shaping the games functionality around it. The game was developed on my personal computer which I built In 2015, it features a intel i5 processor and solid state drive as well as a Nvidia GTX 960 graphics card , all of which when working together are well capable of running a unity 3D environment especially compared to my laptop which is vastly underpowered in comparison.

Architecture for the solution

For this project I used free Assets from the unity asset store and decided to use a low poly art style as it helps with performance. The player is controlled by a character controller component which is accessed in the movementPlayer.cs

script. It applies gravity and checks that the player isGrounded to allow the player to jump. The player's hunger mechanics are held within in this class as in the update method we check if we have 0 hunger which will decide if the player can move.

```
InteractionRay();
Hunger = Hunger - HungerOT * Time.deltaTime;
HungerSlider.value = Hunger / HungerMax;

// check if we have starved
//stios movement and activate UI
if (HungerSlider.value <= 0)
{
    moneyUI.SetActive(false);
    crosshair.SetActive(false);
    Hungerbar.SetActive(false);
    Cursor.lockState = CursorLockMode.None;
    starvationUI.SetActive(true);
    mainCam.transform.localRotation = Quaternion.Euler(0f, 0f, 0f);
}
else//if not move freely
{
    isGrounded = Physics.CheckSphere(groundcheck.position, groundDistance, groundMask);

    if (isGrounded && velocity.y < 0)
    {
        velocity.y = -2f;
    }

    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");

    Vector3 move = transform.right * x + transform.forward * z;

    controller.Move(move * speed * Time.deltaTime);

    if (Input.GetButtonDown("Jump") && isGrounded)
    {
        velocity.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
    }

    velocity.y += gravity * Time.deltaTime;

    controller.Move(velocity * Time.deltaTime);
}
```

Inside the player gameObject there is the camera which the player will see through and can look around with as well as rotate the player to move where they are looking on the x axis.

```
Unity Message | 0 references
void Start()
{
    //locks cursor to middle
    Cursor.lockState = CursorLockMode.Locked;
}

// Update is called once per frame
Unity Message | 0 references
void Update()
{
    float mouseX = Input.GetAxis("Mouse X") * lookSensitivity * Time.deltaTime;
    float mouseY = Input.GetAxis("Mouse Y") * lookSensitivity * Time.deltaTime;

    xRotation -= mouseY;
    xRotation = Mathf.Clamp(xRotation, -90f, 90f);

    transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
    playerBody.Rotate(Vector3.up * mouseX);
}
```

The playerVoiceInteraction.cs script is inside the “Queue” gameObject which is just a simple box collider set as a trigger, Using the GR_OnPhraseRecognized Method When the player enters the trigger the “inQueue” bool is set to true and will allow the Order function to be called with a valuestring passed in as parameters aswell as make the shopkeeper wave.

```
2 references
private void GR_OnPhraseRecognized(PhraseRecognizedEventArgs args)
{
    StringBuilder message = new StringBuilder();
    Debug.Log("recognised phrase");
    SemanticMeaning[] meanings = args.semanticMeanings;


    foreach (SemanticMeaning meaning in meanings)
    {
        string keyValue = meaning.key.Trim();
        string valueString = meaning.values[0].Trim();
        message.Append(" Value : " + valueString + " , KEY: " + keyValue);

        if (inQueue == true)
        {
            switch (keyValue)
            {
                case "order":
                    Debug.Log("ordered");
                    Order(valueString);
                    break;
            }
            Debug.Log(message);
        }
        else
        {
            Debug.Log("Your Order wont be taken unless you are in the Queue");
        }
    }
}

0 Unity Message | 0 references
private void OnTriggerEnter(Collider other)
{
    //vendorUI.SetActive(true);
    waver.SetBool("wave", true);
    Debug.Log("object entered Trigger");
    inQueue = true;
}

0 Unity Message | 0 references
private void OnTriggerExit(Collider other)
{
    //vendorUI.SetActive(false);
    inQueue = false;
    Debug.Log("Your Order wont be taken unless you are in the Queue");
    waver.SetBool("wave", false);
}
```

There is a “foodPrefabs” array that will check the name of the food asked for and will set that prefab as the “foodToMake” which is passed into the “BuyFood” function as a parameter. Buy Food will Instantiate the food on the plate in front of the player. The food on the Menu can be seen behind the shop keeper.



```
private void Order(string order)
{
    Debug.Log("You have ordered " + order);
    switch (order)
    {
        case "hamburger":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "chicken":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "waffles":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "milk":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "pizza":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "coke":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "icecream":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
        case "fries":
            foodToMake = Array.Find(foodPrefabs, foodToMake => foodToMake.name == order);
            break;
    }
    BuyFood(foodToMake);
}

// Update is called once per frame
0 Unity Message | 0 references
void Update()
{
}

1 reference
public int BuyFood(GameObject foodToMake)
{
    if (script.gold >= cost)
    {
        script.Pay();
        Debug.Log("You have enough");
        FindObjectOfType<AudioManager>().Play("deposit");
        Instantiate(foodToMake, foodToSpawn.position, foodToSpawn.rotation);
    }
    else
    {
    }
}
```

The Grammar.xml file will recognise that the user is looking to order something if one of the phrases defined below and will output one of the orderItems

```
<rule id="orderFood">
  <tag>out.order="order"</tag>
  <one-of>
    <item>Can i have</item>
    <item>I would like</item>
    <item>Give me </item>
    <item>Make me</item>
  </one-of>
  <one-of>
    <item>a</item>
    <item>the</item>
    <item>one</item>
  </one-of>
  <ruleref uri="#orderItem" />
  <tag>out.order = rules.latest();</tag>
</rule>

<rule id="orderItem">
  <one-of>
    <item>chicken <tag>out = "chicken";</tag> </item>
    <item>waffles <tag>out = "waffles";</tag> </item>
    <item>milk <tag>out = "milk";</tag> </item>
    <item>ice Cream <tag>out = "icecream";</tag></item>
    <item>pizza <tag>out = "pizza";</tag> </item>
    <item>chips <tag>out = "fries";</tag></item>
    <item>fries <tag>out = "fries";</tag></item>
    <item>soda<tag>out = "coke";</tag> </item>
    <item>coke<tag>out = "coke";</tag> </item>
    <item>Fizzy drink<tag>out = "coke";</tag> </item>
  </one-of>
</rule>
```

The player can pickup coins by colliding with their box collider similarly to Queue. The PickupCoins.cs will access the Money script and increment the players money count based on what value the gameObject is given and then destroy the Money object.



Conclusions & Recommendations

In Conclusion I found making a 3D first person perspective game really interesting and challenging at the same time, it gave me a big appreciation for how much work goes into larger scale triple A title video games of the same style from art, environments, mechanics and sound design. It was tricky at times getting to work with mechanics unfamiliar to me in Unity development however it was rewarding to have it all the parts I had working and come together in the end. I do believe that I could have added more in terms of a challenge for the player to maneuver around however unseen problems in

developing a game type that I hadn't touched before cut into my already busy packed college schedule.

I believe that a 3D first person game would be better suited to a team or pairing of developers who could work through the volume of environment setup and design which is quite time consuming. I do believe that my time making games in Unity previously has had a good impact on my understanding of how not only games but how programs can communicate with one another through a visual and interactive lens.

References

https://www.youtube.com/watch?v=NcOv-Rk1RZA&list=LL&index=11&ab_channel=Lurony

https://www.youtube.com/watch?v=QairabyTJc&list=LL&index=17&ab_channel=Brackeys

https://www.youtube.com/watch?v=86ilSrawK98&list=LL&index=9&t=128s&ab_channel=Janyanam

https://www.youtube.com/watch?v=Zb9WchxZhvM&list=LL&index=10&t=318s&ab_channel=JasonWeimann

https://www.youtube.com/watch?v=JtflOvhOO1Y&ab_channel=MattWester

Links

<https://github.com/RobertDonnelly/LateNightSnack-GBUI-Project>