DOCUMENTACIÓN POLYMON AR

Estructura del programa y toma de decisiones: El programa esta estructurado de manera que tenemos un main en el que se realiza toda la lógica de decisiones y donde se muestra el frame de la camara. Luego tenemos varios archivos en los que cada uno realiza las distintas tareas asignadas a cada una de las siguientes funcionalidades:

- Autenticación mediante reconocimiento facial: El codigo para realizar esto se encuentra en reconocer_caras.py y el como almacenar el usuario en user_data.py. Para reconocer caras he usado el modulo face_recognition y para almacenar al usuario he creado una clase UserData que se apoya en un json con todos los usuarios registrados y que tiene las funciones necesarias para gestionar todo lo relacionado con los usuarios como añadir uno nuevo, añadir un poligono a un cierto usuario…

- Interacción mediante voz: El codigo para esto se encuentra en voice_recognition.py, text_voice.py, traducciones.py, Para la interacción por voz he empleado speech_recognition con el reconocedor de google para que el usuario pueda hablar con el ordenador y el modulo pyttsx3 para que el ordenador pueda hablar. Para permitir esto he creado dos hilos que funcionan como demonios ejecutando todo el rato una función en la que se comprueba si el evento asociado a hablar o a escuchar esta activo en el caso del reconocedor de voz y para el caso de hablar he usado una cola a la cual le voy metiendo las frases que quiero y desde el hilo que habla, cuando detecta que la cola no esta vacía, comienza a hablar. El programa soporta dos idiomas ingles y español y para esto he creado dos colas de idiomas. Tanto para escuchar como hablar estoy obteniendo todo el rato el idioma del usuario el cual se lo paso a las dos colas de idiomas y a la hora de escuchar y de hablar empleara el idioma del usuario.

- Deteccion de poligonos: El código para esto se encuentra en reconocer_poligonos.py. Para la detección de poligonos he decidido hacerlo en un solo frame cada vez ya que es demasiado costoso y ralentiza demasiado el mostrar frames. He decidio usar las funciones GaussianBlur y el algoritmo Canny porque es lo que mejores resultados me ha dado a la hora de detectar poligonos. Además he ajustado en función del area del polígono detectado para descartar el mayor número posible de falsos poligonos. Los posibles poligonos a identificar van desde lado 3 a lado 12. Podríamos añadir más simplemente poniendo el número de lados del poligono y el nombre a la hora de detectar poligonos.

- Pintar polígonos: El código para esto se encuentra en pintar_poligonos.py. Para pintar polígonos empleo un marcador de aruco como referencia. A partir de éste calculo la circunferencia circunscrita tomando como radio el vector desde el centro hasta una esquina y luego pongo los vertices sobre esa circunferencia en función del número de lados del poligono a pintar. Para rellenarlo y pintar solo las aristas visibles uso el punto de vista de la camara. También escribo el nombre del poligono que se usa como base en la cara superior del prisma o el vértice de la pirámide. Para realizarlo lo más realista posible también pense en calcular el plano que contiene al marcador y proyectar los puntos de nuestro poligono en dicho marcador pero no me producía el resultado deseado.

- Almacenamiento de datos: Puesto que necesito almacenar los datos de los usuarios y las posibles traducciones por si se quisiera añadir más idiomas necesito de una estructura para almacenarlo y he elegido json para mediante clave valor ir usando luego en el programa lo que me iba haciendo falta.

FUNCIONAMIENTO DE LA APLICACIÓN

Aclaración: Todas las interacciones son mediante voz, tanto el ordenador como lo que desea transmitir el usuario, excepto la galería que simplemente muestra los nombres de los poligonos almacenados.
Comenzamos sin usuario por lo que nos tendremos que registrar. Tras esto se nos pide primero el idioma que admite tanto español como inglés. Con esto cargaremos las traducciones en función del idioma elegido y a partir de aqui nos hablará siempre en el idioma que le hemos dicho. Después nos pide el nombre con el cuál completaremos el registro de usuario y pasaremos a estar con la sesión iniciada.
Tanto el idioma como el nombre tienen tolerancia a errores por lo que si no nos da tiempo o decimos algo sin sentido nos lo pedirá de nuevo. Para borrar un usuario registrado debemos eliminarlo del fichero.
Una vez tenemos nuestro usuario creado se nos pedirá que queremos hacer si detectar poligonos, pintar poligonos o ver la galeria de poligonos. Un usuario recien creado no tiene polígonos almacenados por lo que debemos comenzar detectando poligonos. Para detectar un poligono debemos enseñar a la camara el polígono que queramos almacenar(y que se pueda registrar es decir de 3 a 12 lados) y si todo es correcto se almacenará el poligono.
Tras esto nos volverá a pedir que deseamos hacer y ya si podemos pintar o mostrar la galería. Si elegimos ver la galería simplemente nos sacará por pantalla los polígonos que tiene almacenados el usuario. Si elegimos pintar deberemos mostrarle el aruco sobre el cual se nos pintará el prisma con base el primero polígono identificado por el usuario. Podemos manejar varias situaciones mientras que estamos pintando que son las siguientes:
-Si decimos “color y el color que queremos” comprobará si el color que queremos esta disponible y al polígono que esta pintando le pondrá ese color.
-Si decimos “altura y la altura que queremos” si la altura que hemos dicho es valida le pondrá a la figúra dicha altura.
-Si decimos “prisma” o “piramide” alternará entre pintar el prisma de base el poligono y la pirámide de base el poligono.
-Si decimos “siguiente” cambiará al siguiente polígono.
-Si decimos salir volverá al menú anterior en el que nos preguntará si queremos detectar, pintar o ver la galería.
Para salir de la aplicación simplemente pulsando q se nos cerrará por completo.

DEPENDENCIAS

He usado las librerias face_recognition, opencv, pyttsx3, math, numpy, json y os.