UNIVERSIDAD
DE GRANADA

QuimicAR: Realidad Aumentada en la enseñanza de la Química

CUIA - Curso 2023/2024

Autor

Pablo Luque Salguero

Granada, 23 de junio de 2024

Índice

1 | Descripción de QuimicAR

QuimicAR se plantea como una herramienta de apoyo al aprendizaje en el área de la química básica. Su función es la de introducir el concepto de *moléculas* desde la realidad aumentada. El usuario es capaz de interactuar con *átomos* y observar cómo es posible combinarlos para crear *moléculas*. La aplicación se compone por una serie de puzles que el usuario debe resolver, encontrando la combinación correcta de *átomos* para formar una determinada *molécula*.

2 | Tecnologías involucradas

2.1 | Realidad aumentada

El punto central de la realidad aumentada son los marcadores ArUco. Estos marcadores son impresos en papel y colocados en una superficie plana. La cámara del dispositivo captura la imagen de los marcadores y la aplicación superpone los modelos tridimensionales correspondientes.

En cada marcador, se muestra un grupo de átomos o moléculas. Los átomos son esferas de colores, mientras que las moléculas son combinaciones de átomos. El usuario puede interactuar con los átomos y moléculas, moviendo y rotando el marcador en el espacio. La aplicación detecta los movimientos del marcador y actualiza la posición de los átomos y moléculas en tiempo real.

Para la detección de marcadores, se utiliza la librería OpenCV [1]. OpenCV es una librería de visión computacional que provee una serie de algoritmos para el procesamiento de imágenes y videos. La detección de marcadores se realiza en tiempo real, utilizando la cámara del dispositivo.

Para la representación tridimensional, se utiliza OpenGL a través de una librería de Python llamada ModernGL [2]. Toda la aplicación se ejecuta en el contexto de una ventana de OpenGL. En este contexto se combinan un rectángulo del tamaño de la pantalla al que se le aplica una textura con la imagen que se obtiene de la cámara y los modelos tridimensionales que se superponen encima de los marcadores.

El cambio del punto de vista de la cámara de OpenCV a OpenGL no es un problema trivial, pues los sistemas de coordenadas de ambas librerías son diferentes. También se debe tener en cuenta la forma correcta de obtener una matriz de proyección correcta a través de las características intrínsecas de la cámara.

2.2 | Reconocimiento de voz

El reconocimiento de voz se utiliza para la interacción explícita con la aplicación. La aplicación se comunica con el usuario a través de mensajes de voz y el usuario puede interactuar con la aplicación a través de comandos de voz. En particular, estos comandos de voz se utilizan para avanzar al siguiente puzle una vez el usuario haya superado el puzle actual.

Para el reconocimiento de voz, se utiliza la librería de Python SpeechRecognition [3]. SpeechRecognition es una librería de Python que provee una interfaz sencilla para trabajar con motores de reconocimiento de voz. La librería soporta varios motores de reconocimiento de voz, entre ellos Google Speech Recognition, Sphinx y Wit.ai. En mi caso estoy usando el motor de Google, pues presenta un buen rendimiento, es gratuito, y no requiere descargar un modelo local.

Para que la aplicación produzca mensajes de voz, se utiliza la librería Pyttsx3 [4]. Con esta librería se pueden generar mensajes de voz a partir de texto.

Para ambas tareas, el reconocer la voz del usuario y generar mensajes de voz, es necesario que se ejecuten en procesos separados. Esto es así porque ambas acciones son procesos bloqueantes, y si se ejecutara en el mismo hilo que el programa principal, la aplicación no podría actualizar la interfaz gráfica.

2.3 | Consciencia de contexto

La aplicación tiene consciencia sobre el usuario que está interactuando con ella. La primera vez que un usuario interactúa con la aplicación, se almacena una codificación de su rostro. Esto se realiza con la librería de Python face_recognition [5]. Esta librería permite identificar rostros en una imagen, y también permite codificarlos en una serie de números que representan las características del mismo.

3 | Manual de usuario

Esta aplicación utliza el diccionario de marcadores 6x6 de ArUco, por lo que es necesario imprimir al menos 4 marcadores para poder utilizar la aplicación.

La primera vez que se ejecuta la aplicación, esta intenta identificar al usuario, y le pedirá un nombre para registrarlo. Para la comunicación, la aplicación utiliza el inglés. Una vez que el usuario se ha registrado, se carga el primer puzle. A partir de aquí se recomienda que la cámara apunte a una superficie plana, con una buena iluminación.

En la interfaz que ve el usuario aparece en la esquina superior izquierda la molécula objetivo del puzle, junto con el número de marcadores que son necesarios para mostrar todos los átomos. No todos los átomos serán necesarios para resolver el puzle, por lo que el usuario deberá identificar cuáles son los átomos que le permitirán resolver el puzle.

Una vez identificados los átomos necesarios, el usuario puede agruparlos físicamente al acercarlos. La aplicación detectará la agrupación y si es correcta, generará la molécula objetivo. Si la agrupación es incorrecta, la aplicación no generará nada. La molécula resultante puede ser separada al alejar los marcadores entre sí.

Cuando se encuentra la solución, la aplicación pregunta al usuario si quiere pasar al siguiente puzle. Para ello, el usuario deberá decir “yes” o “no”. Si el usuario dice “yes”, la aplicación cargará el siguiente puzle. Si el usuario dice “no”, la aplicación seguirá mostrando el puzle actual. Esta pregunta se hará cada vez que se combine la molécula objetivo.

Cuando el usuario haya completado todos los puzles, la aplicación le felicitará y volverá a comenzar desde el primer puzle.

4 | Instalación

En este apartado se detalla el proceso de instalación de las dependencias necesarias para ejecutar la aplicación. La aplicación ha sido desarrollada en Python 3.11, por lo que es necesario tenerlo instalado. Todo el código fuente de la aplicación se encuentra en el repositorio de Github https://github.com/pluque01/chemistry-augmented-reality.

4.1 | Usando venv

Venv es un módulo de Python que permite crear entornos virtuales. Para instalar la aplicación, se pueden seguir los siguientes pasos:

1. `git clone https://github.com/pluque01/chemistry-augmented-reality.git QuimicAR && cd QuimicAR`
2. `python -m venv venv`
3. `source venv/bin/activate` (bash/zsh) o `venv\Scripts\activate.bat` (cmd)
4. `pip install -r requirements.txt`
5. `python ./chemistry_ar/chemistry-ar.py`

4.2 | Usando Poetry

Poetry es una herramienta para manejar dependencias en proyectos de Python. Este proyecto utiliza esta herramienta, pero no es necesaria para instalar la aplicación. Sin embargo, si ya se tiene instalada, se pueden ejecutar los siguientes comandos:

1. `git clone https://github.com/pluque01/chemistry-augmented-reality.git QuimicAR && cd QuimicAR`
2. `poetry install`
3. `poetry run python ./chemistry_ar/chemistry-ar.py`

5 | Referencias

[1] Gary Bradski. The opencv library. https://opencv.org/, 2000. Versión 4.9.

[2] Martin Czaki. Moderngl. https://github.com/moderngl/moderngl, 2024. Versión 5.10.0.

[3] Anthony Zhang. Speechrecognition library. https://github.com/Uberi/speech_recognition, 2017. Versión 3.10.4.

[4] Jem Finer. pyttsx3 library. https://pypi.org/project/pyttsx3/, 2017. Versión 2.90.

[5] Adam Geitgey. face_recognition library. https://github.com/ageitgey/face_recognition, 2017. Versión 1.3.0.