

Reporte de Proyecto Grupal 3

Aplicación de Seguimiento Facial de Realidad Aumentada

González Reyes Carmen Rubí*, Guerrero Salazar Julissa Vianney*, Hernández Rodríguez Elías*, Martínez Ibarra César Eduardo* y Resendiz García Alex Emil*
Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria

Resumen—Este proyecto de Realidad Aumentada para Android se desarrolló integrando ARCore y Sceneform para permitir la superposición de modelos 3D en rostros detectados. El proceso de desarrollo abarcó desde la configuración del entorno y la implementación técnica del seguimiento de rostros en tiempo real y la carga asíncrona de modelos en MainActivity.java, hasta la creación de una interfaz de usuario intuitiva para la selección de modelos en ModelSeleccionMenu.java y ModelMapAdapter.java. Como parte del ciclo de desarrollo de la aplicación de Realidad Aumentada que interactúa con hardware y software complejos, se realizaron pruebas exhaustivas en dispositivos móviles para verificar la correcta detección y seguimiento de rostros, la superposición precisa de modelos 3D y la funcionalidad del menú de selección.

I. INTRODUCCIÓN

En un mundo digital en constante evolución, la Realidad Aumentada (RA) se ha establecido como una herramienta versátil [1]. Sin embargo, su integración en dispositivos móviles a menudo presenta desafíos, desde la complejidad de la configuración técnica hasta la implementación de un seguimiento facial preciso [2].

La aplicación de la RA y el seguimiento facial es un campo de estudio activo, con diversos trabajos que exploran su potencial en distintos sectores. En el ámbito del transporte, se han desarrollado sistemas que utilizan la RA para superponer datos e indicadores directamente en el entorno[3]. De manera similar, en el sector educativo, se han implementado plataformas que emplean el reconocimiento facial para la gestión y el seguimiento de estudiantes, mejorando la seguridad y el control de asistencia[4].

Este proyecto presenta una aplicación de RA para móviles, desarrollada en el entorno de Android Studio[5, 6], utilizando Java. La aplicación se basa en la integración de las bibliotecas ARCore[7] y Sceneform[8], permitiendo la superposición de modelos 3D sobre rostros detectados en tiempo real a través de la cámara frontal del dispositivo[9, 10].

II. DESARROLLO EXPERIMENTAL

II-A. Fase 1: Configuración de la Arquitectura y Entorno

El viaje de desarrollo de esta aplicación de Realidad Aumentada (RA) para Android comenzó con la configuración fundamental del proyecto. En el entorno de desarrollo Android

Studio (AS), se sentaron las bases integrando las dependencias esenciales de ARCore y Sceneform en el archivo **build.gradle**. Esto fue crucial, ya que estas bibliotecas otorgan acceso a las interfaces de programación de aplicaciones necesarias para habilitar tanto la RA como la capacidad de renderizar modelos 3D. Paralelamente, se ajustó el **AndroidManifest.xml** para solicitar los permisos de cámara indispensables y declarar la compatibilidad con las características de ARCore, asegurando así que la aplicación pudiera interactuar correctamente con el hardware del dispositivo y la plataforma de RA. La organización de los recursos también fue un paso temprano y vital; se crearon las estructuras de carpetas **assets/models** y **assets/texturas** para almacenar los archivos de modelos 3D en formato **.gib** y sus texturas asociadas, preparando el escenario para la carga de activos visuales.

II-B. Fase 2: Implementación del Core de RA (MainActivity.java)

Con la arquitectura establecida, el foco se trasladó a la implementación del núcleo de la experiencia de RA, principalmente dentro de **MainActivity.java**. Aquí, se inicializó la vista de RA obteniendo o añadiendo dinámicamente un **ArFrontFacingFragment** al diseño de la actividad. Este fragmento, un componente clave de Sceneform UX, asume la responsabilidad de gestionar la cámara frontal y orquestar la sesión de ArCore específicamente para el seguimiento de rostros. Una vez que el fragmento estuvo operativo, su método **onViewCreated** proporcionó acceso a la **ArSceneView**. En este punto, se ajustó la prioridad del renderizado de la transmisión de la cámara y, de manera fundamental, se registró un **setOnArgumentedFaceUpdateListener**. Este oyente actúa como el conducto a través del cual ARCore comunica los eventos de detección de rostros o cambios en su estado de seguimiento, activando respuestas dentro de la aplicación. La carga de los modelos 3D se manejó de forma asíncrona utilizando **ModelRenderable.builder**, lo que permitió que la aplicación continuara respondiendo mientras los activos se cargaban en segundo plano. El uso de **CompletableFuture** facilitó la gestión de estas operaciones, asegurando que el modelo renderizado (faceModel) estuviera disponible para

su uso una vez completada la carga y proporcionando un mecanismo para manejar posibles errores durante el proceso.

La gestión del seguimiento de rostros orquestada por el método **onArgumentedFaceTrackingUpdate**, es donde la RA cobra vida. Cuando ARCore detecta un **AugmentedFace** con un estado **TRACKING**, la aplicación responde creando una instancia de **AugmentedFaceNode**, un nodo de Sceneform especializado diseñado para integrarse con datos de rostros aumentados. A este nodo se le asigna el **faceModel** cargado previamente utilizando **setFaceRegionRenderable**, preparando el modelo 3D para ser superpuesto en el rostro detectado. Se configuraron propiedades como **setShadowCaster** y **setShadowReceiver** para refinar el comportamiento de las sombras del modelo en la escena. Posteriormente, el **AugmentedFaceNode** se añade a la escena de la **ArSceneView**, integrando visualmente el modelo 3D con la transmisión de la cámara frontal. Para mantener un registro de los rostros rastreados y sus nodos correspondientes. Para mantener un registro de los rostros rastreados y sus nodos correspondientes, se utilizó un **HashMap**, mapeando cada **AugmentedFace** a su **AugmentedFaceNode** asociado para una fácil referencia y manipulación. Si el estado de seguimiento de un rostro cambia a **STOPPED**, el **AugmentedFaceNode** correspondiente se elimina cuidadosamente de la escena y del **HashMap**, liberando recursos y manteniendo la escena limpia. Adicionalmente, se incorporó la reproducción de audio mediante un **MediaPlayer**, permitiendo que un archivo de sonido específico se reprodujera condicionalmente cuando se seleccionara un modelo particular, añadiendo una capa extra de interactividad a la experiencia de RA.

II-C. Fase 3: Desarrollo del Menú de Selección de Modelos (ModelSeleccionMenu.java y ModelMapAdapter.java)

Paralelamente a la implementación del núcleo de RA, se desarrolló la funcionalidad del menú de selección de modelos, que abarca **ModelSeleccionMenu.java** y **ModelMapAdapter.java**. La actividad **ModelSelectionMenu.java** fue diseñada para presentar a los usuarios las opciones de modelos 3D disponibles a través de un **GridView**. Se definió un **LinkedHashMap** (`modelList`) para almacenar la asociación entre los nombres de visualización de los modelos y los nombres de archivo **.glb** correspondientes, manteniendo el orden de las opciones. Se adjuntó un **OnItemClickListener** al **GridView**, permitiendo que la aplicación respondiera cuando un usuario seleccionara un modelo. Tras la selección, se obtuvo el nombre del archivo del modelo elegido y se comunicó de vuelta al **MainActivity** utilizando un **Intent**, señalando el resultado de la selección y finalizando la actividad del menú. Para gestionar la visualización de los modelos en el **GridView**, se creó un adaptador personalizado, **ModelMapAdapter.java**, que extiende **BaseAdapter**. En su método **getView()**, este adaptador se encarga de inflar un diseño (**model_item.xml**) para cada elemento del **GridView** asignando el nombre del modelo a un **TextView** dentro de ese diseño para su visualización.

En conjunto, este proceso de desarrollo entrelaza la configuración del entorno, la implementación técnica del seguimiento de rostros, la superposición de modelos 3D, y la creación de una interfaz de usuario para la selección de modelos. Cada archivo de código desempeña un papel específico y se interconecta con los demás a través de mecanismos como Intents y listeners de eventos, culminando en una aplicación de RA funcional que permite a los usuarios superponer modelos 3D en rostros detectados utilizando la cámara frontal del dispositivo.

III. RESULTADOS

El proyecto culminó en una aplicación funcional que demuestra cómo la realidad aumentada puede interactuar con el entorno del usuario. Los resultados se organizaron en tres áreas clave: la integración de los modelos 3D, el funcionamiento del menú de selección y la efectividad del seguimiento facial.

III-A. Modelos 3D y su integración

Se incluyeron varios modelos 3D en la aplicación. Estos modelos se cargaban de forma asíncrona en la aplicación. Se incluyeron desde máscaras simples hasta elementos más elaborados, como gafas y sombreros. El uso de propiedades como **setShadowCaster** y **setShadowReceiver** para que los modelos se vieran realistas, interactuando con la luz del entorno. Esto ayudó a que los modelos se integraran de manera creíble en la escena.

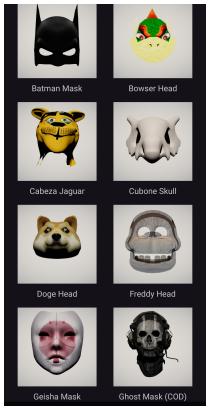


(a) Máscara de esqueleto
 (b) Cabeza de mario
 (c) Casco vintage

Figura 1: Modelos 3D superpuestos en un rostro. La figura (a) muestra una máscara simple, (b) una cabeza de un personaje y (c) un casco. Estos ejemplos demuestran la variedad de elementos que la aplicación puede renderizar.

III-B. Menú de selección

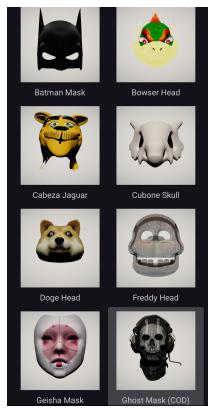
El menú de selección se diseñó para que fuera fácil de usar. Cuando el usuario selecciona un modelo, la aplicación envía esta información a la pantalla principal para que el modelo se aplique inmediatamente. El resultado fue una navegación simple e intuitiva para el usuario.



(a) Menú de modelos



(b) Menú de modelos 2



(c) Selección de un modelo

Figura 2: Interfaz del menú de selección de modelos. La figura (a) muestra el menú principal, (b) la continuación de la lista de modelos y (c) la selección de un modelo.

III-C. Seguimiento de rostros y superposición de RA

La aplicación usa la cámara frontal para detectar un rostro y, una vez encontrado, lo sigue de manera precisa. El sistema puede seguir la rotación y posición con precisión. Si un rostro deja de verse, el modelo 3D asociado desaparece, liberando recursos.



(a) De frente



(b) De lado



(c) Mirando arriba

Figura 3: Ejemplos del seguimiento de rostros en tiempo real. La figura (a) muestra la detección inicial, (b) el seguimiento con movimiento de cabeza y (c) el seguimiento con otro movimiento.

IV. CONCLUSIÓN

En conclusión, el desarrollo de esta aplicación de Realidad Aumentada para Android, que permite superponer modelos 3D en rostros detectados, demuestra una integración exitosa de ARCore y Sceneform. El proceso abarcó desde la configuración técnica inicial y la implementación del seguimiento de rostros en tiempo real y la carga asíncrona de modelos, hasta la creación de una interfaz de usuario para la selección

de modelos. La interconexión de archivos java a través de mecanismos como listeners e intents fue fundamental para lograr una experiencia de Realidad Aumentada funcional y reactiva, ofreciendo al usuario la capacidad de interactuar con modelos 3D superpuestos en sus rostros utilizando la cámara frontal del dispositivo.

REFERENCIAS

- [1] Javier Fombona Cadavieco, María Ángeles Pascual Sevillano y María Filomena Madeira Ferreira Amador. “Realidad aumentada, una evolución de las aplicaciones de los dispositivos móviles”. En: *Pixel-Bit. Revista de medios y educación* 41 (2012), págs. 197-210.
- [2] Alex Yuri Bernal Leyva. “Análisis de Métodos de Reconocimiento Facial bajo el Sistema Operativo Android”. En: (2018).
- [3] Cesar Jaime Montiel Moctezuma et al. “Implementación de algoritmos de realidad aumentada para la visualización de indicadores en el autotransporte”. En: () .
- [4] Juan Manuel Castro Florez, Javier Enrique Marrugo Cogollo et al. “Sistema de reconocimiento facial para la gestión y el seguimiento de estudiantes ausentes (Sefad)”. En: (2022).
- [5] Android Developers. *Android Developers Blog*. <https://android-developers.googleblog.com/>. Consultado en agosto de 2025. 2025.
- [6] Bill Phillips, Chris Stewart y Kristin Marsicano. *Android Programming: The Big Nerd Ranch Guide*. 4.^a ed. Big Nerd Ranch Guides, 2019. ISBN: 9780134706054.
- [7] ARCore. *Descripción general de ARCore y los entornos de desarrollo compatibles*. <https://developers.google.com/ar/develop?hl=es-419>. Consultado en agosto de 2025. 2025.
- [8] sceneform. *Descripción general de la forma de escena*. <https://developers.google.com/sceneform/develop?hl=es-419>. Consultado en agosto de 2025. 2025.
- [9] Google for Developers. *Guía de detección de puntos de referencia facial para Android*. https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker/android?hl=es-419. Consultado en agosto de 2025. 2025.
- [10] OurTechArt. *Augmented Reality Virtual Reality*. <https://www.ourtechart.com/product/paparmali-2-ar-face-tracking-using-kinect-2/>. Consultado en agosto de 2025. 2025.