Manual Técnico

Proyecto Final de Realidad Aumentada

Dental AR

Elaborado por:

Flores Cruz Daniel

Fecha de entrega:

16 de mayo de 2025

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Índice

1. Introducción

El presente manual técnico documenta de forma integral el desarrollo del proyecto **Dental AR**, una aplicación interactiva diseñada para dispositivos móviles con soporte de realidad aumentada. Este proyecto fue concebido con el objetivo de facilitar el aprendizaje de la salud bucal infantil mediante una plataforma lúdica, educativa y accesible.

A lo largo de este documento se detallan los aspectos técnicos, estructurales y funcionales que conforman Dental AR, desde la arquitectura del sistema y la integración de motores AR, hasta la lógica de los minijuegos y la organización de la interfaz de usuario. El manual también proporciona lineamientos para su mantenimiento, ampliación futura y portabilidad.

2. Objetivo del Proyecto

Dental AR tiene como meta principal crear un entorno interactivo donde los niños puedan aprender sobre higiene bucal de una manera visual, didáctica y atractiva. La aplicación está compuesta por dos modos principales: uno educativo y otro basado en retos gamificados. En ambos casos, el usuario puede interactuar con modelos 3D de dientes, identificar sus nombres y funciones, así como simular procedimientos como la eliminación de caries o la extracción de muelas.

3. Tecnologías Utilizadas

- **Motor de desarrollo:** Unity 2021.3.X LTS
- **Lenguaje de programación:** C#
- **Motor de Realidad Aumentada:** Vuforia Engine
- **Sistema operativo de desarrollo:** macOS Sonoma (MacBook Pro M3 Pro)
- **Dispositivo objetivo:** iPhone 12 Pro Max con soporte para ARKit
- **Control de versiones:** Git + GitHub
- **Diseño gráfico:** Adobe Illustrator, Photoshop y DALL · E
- **Documentación:** LaTeX

4. Estructura del Proyecto

4.1. Organización General de la Escena (Unity)

- **DentalAR (Scene Root)**
  - Directional Light
  - ARCamera
  - ImageTarget
    - LearnContent
    - ChallengeObject (Reto 1)
    - Challenge2Object (Reto 2)
    - Trofeo
  - EventSystem
  - Menú Principal
  - Menú Aprende
  - Menú Principal Retos
  - Menú Reto1 Caries
  - UIManager

5. Procedimientos Técnicos

Esta sección detalla los pasos técnicos realizados durante la implementación del proyecto Dental AR, desde la configuración del entorno de desarrollo hasta la estructura funcional interna del sistema.

5.1. Integración de Vuforia en Unity

La integración del motor de Realidad Aumentada Vuforia con Unity fue uno de los primeros pasos fundamentales en el desarrollo. El procedimiento seguido fue:

1. Instalación de Unity 2021.3.X LTS mediante Unity Hub.
2. Activación de Vuforia desde el menú Edit / Project Settings / XR Plug-in Management.
3. Registro y obtención de licencia gratuita en developer.vuforia.com.
4. Configuración de la licencia desde Vuforia Configuration.
5. Agregado de ARCamera e ImageTarget a la escena.
6. Importación del paquete .unitypackage desde Vuforia Target Manager.

5.2. Preparación de Modelos 3D

Los modelos tridimensionales utilizados en la aplicación fueron preparados de forma que permitieran interacciones precisas y pedagógicas. El flujo de trabajo fue:

- Preparacion de modelos en Blneder , donde se cambio su configuracion de texturas por medio de generacion de image texrtures con unicamente color lo que permitiria compatibilidad con la stexturas en Unity.
- División en submodelos para permitir interacciones individuales.
- Ajuste de escala, rotación y materiales.
- A cada objeto interactivo se le asignó un BoxCollider para detección de colisiones, y un script específico para gestionar su respuesta a la interacción táctil.
- Los modelos se organizaron dentro de contenedores lógicos (LearnContent, ChallengeObject, Challenge2Object) con el fin de facilitar su activación/desactivación dinámica en cada modo.

5.3. Lógica de Interacción (Scripts)

El comportamiento dinámico del sistema fue desarrollado mediante scripts en C, diseñados bajo principios de modularidad y separación de responsabilidades. Las acciones clave incluyeron:

- Uso de OnMouseDown() o Raycasting para capturar interacciones del usuario sobre objetos 3D.
- ToothInfoSingle.cs: despliega información al tocar un diente en modo Aprender.
- QuitarCaries.cs: desactiva el objeto al ser tocado y notifica el avance al ManagerRetoDental.cs.
- AnestesiaPaso.cs, AflojarPaso.cs, y ExtraerPaso.cs: definen la secuencia lógica del reto de extracción dental.
- UIManager.cs: centraliza la navegación entre menús, activación de objetos y gestión de eventos visuales.

  Cada script fue diseñado para ser reutilizable, con enfoque en claridad, control de estado y escalabilidad.

5.4. Gestión de Escenas y UI

El proyecto cuenta con una única escena base (DentalAR), pero la navegación interna entre modos y menús se maneja mediante activación y desactivación de objetos de interfaz (Canvas). La implementación se realizó de la siguiente manera:

- Se crearon distintos Canvas para cada sección: menú principal, modo aprender, menú de retos, reto de caries, y pantallas de victoria.
- Cada botón se conectó a funciones públicas dentro de UIManager.cs usando onClick.AddListener().
- Se utilizaron métodos como SetActive(true/false) para mostrar u ocultar secciones según el contexto actual.
- Se implementó una función ResetearReto() que limpia el estado anterior del minijuego antes de iniciar uno nuevo.
- El ImageTarget se mantiene activo en todo momento para asegurar que los modelos AR estén correctamente anclados.

5.5. Adaptación a Dispositivo Móvil iOS

La aplicación fue diseñada para ejecutarse en dispositivos móviles de Apple, específicamente en un iPhone 12 Pro Max con sistema operativo iOS y soporte completo para tecnologías de Realidad Aumentada mediante ARKit.

El procedimiento de configuración y despliegue fue el siguiente:

1. En File ¿Build Settings, se seleccionó iOS como plataforma de destino en Unity.
2. Se instaló Xcode en el sistema de desarrollo (macOS Sonoma) y se configuró un perfil de desarrollador de Apple.
3. Se generó un proyecto Xcode desde Unity utilizando la opción Build, lo cual creó los archivos necesarios para compilación en dispositivos iOS.
4. Desde Xcode, se configuraron los permisos de cámara, se ajustó la orientación de la interfaz a modo vertical (Portrait) y se seleccionó el dispositivo de prueba (iPhone 12 Pro Max).
5. Se conectó el iPhone físicamente a la Mac mediante cable USB y se ejecutó el proyecto desde Xcode utilizando Run, lo cual instaló la app en el dispositivo.
6. Se realizaron pruebas en campo para validar la correcta ejecución de AR, la estabilidad del renderizado y el desempeño general de la aplicación bajo las condiciones reales de uso.

Gracias al soporte nativo de ARKit en dispositivos iOS y la integración fluida de Unity con Xcode, la aplicación se comportó de forma óptima en el entorno de destino.

5.6. Control de Versiones y Repositorio

El control de versiones fue una parte integral del desarrollo, permitiendo mantener la trazabilidad de los cambios, evitar conflictos y respaldar el avance del proyecto.

- Se utilizó Git como sistema de control de versiones, con repositorio privado alojado en GitHub.
- Se configuró un archivo .gitignore específico para Unity, excluyendo archivos temporales, bibliotecas cacheadas y configuraciones de plataforma.
- El repositorio se estructuró en carpetas principales: /Assets: todos los elementos del proyecto (modelos, scripts, prefabs, escenas). /Docs: documentos técnicos y reportes del proyecto. /Build: versiones exportadas del APK. README.md: instrucciones básicas de uso y despliegue.

6. Modos de Juego

6.1. Modo Aprender

Este modo ofrece un recorrido visual y educativo sobre los diferentes tipos de dientes. Se accede mediante el botón “Aprender” en el menú principal. Al activarse, se oculta la interfaz inicial y se despliega LearnContent, que presenta un conjunto de modelos 3D que representan cada tipo de diente humano (incisivos, caninos, premolares, molares y muelas del juicio).

Cada diente tiene asignado un script (ToothInfoSingle.cs) que detecta el toque del usuario mediante raycasting o interacción directa (OnMouseDown) y despliega información textual o visual relacionada. Este modo tiene fines exclusivamente didácticos y no incluye desafíos ni gamificación.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ToothInfoSingle : MonoBehaviour
{
    [Header("Modelo de muela")]
    public GameObject toothModel;       // El modelo 3D de la
       muela
    public GameObject objectToShow;     // El objeto 3D que
       aparecer  al hacer clic

    void Start()
    {
        // Aseg rate de que el objeto est  oculto al inicio
        if (objectToShow != null)
            objectToShow.SetActive(false);
    }

    void OnMouseDown()
    {
        // Verifica que el objeto no sea nulo
        if (objectToShow != null)
        {
            // Alterna la visibilidad del objeto 3D al hacer clic
            bool isActive = objectToShow.activeSelf;
            objectToShow.SetActive(!isActive);
        }
        else
        {
            Debug.LogWarning("No se ha asignado el objeto a
               mostrar.");
        }
    }
}
```

Listing 1: Script ToothInfoSingle.cs

6.2. Modo Reto 1: Quitar Caries

Este reto se activa desde el botón “Reto1 Caries” dentro del menú de retos. Utiliza como contenedor principal el objeto ChallengeObject, que incluye un modelo de dientes con varias caries visibles como GameObjects.

Cada carie está controlada por el script QuitarCaries.cs, que al detectar un toque, desactiva la carie y notifica al ManagerRetoDental.cs, quien lleva un conteo del progreso del reto. Al eliminarse las siete caries correctamente, se activa el objeto trofeo y se despliega la interfaz VictoriaUI, generando una retroalimentación positiva al usuario.

Codigo:

```
using UnityEngine;

public class QuitarCaries : MonoBehaviour
{
    [Header("Efectos opcionales")]
    public ParticleSystem efectoDestruccion;
    public AudioClip sonidoDestruccion;

    [Header("Referencias")]
    public ManagerRetoDental managerRetoDental;

    private bool yaFueEliminada = false;

    void OnMouseDown()
    {
        if (!Application.isPlaying || yaFueEliminada) return;

        yaFueEliminada = true;

        // Efectos visuales y sonoros
        if (efectoDestruccion != null)
            Instantiate(efectoDestruccion, transform.position,
               Quaternion.identity);

        if (sonidoDestruccion != null)
            AudioSource.PlayClipAtPoint(sonidoDestruccion,
               transform.position);

        // Ocultar SOLO este objeto (la carie)
        gameObject.SetActive(false);

```

```
30          // Notificar al manager
31          if (managerRetoDental != null)
32              managerRetoDental.CarieEliminada();
33          else
34              Debug.LogError("ManagerRetoDental no asignado en " +
                    gameObject.name);
35      }
36  }
```

Listing 2: Script QuitarCaries.cs

```
1   using UnityEngine;
2   using System.Collections.Generic;
3
4   public class ManagerRetoDental : MonoBehaviour
5   {
6       [Header("Caries del reto")]
7       public List<QuitarCaries> caries; // Asigna los 7 objetos
            caries con este script
8       private int contadorCaries = 0;
9
10      [Header("Referencias")]
11      public GameObject modeloExito;        // Trofeo
12      public UIManager uiManager;           // Manejador de UI
13
14      public void ResetearReto()
15      {
16          contadorCaries = 0;
17
18          foreach (var carie in caries)
19          {
20              if (carie != null)
21              {
22                  // Reactivar la carie
23                  carie.gameObject.SetActive(true);
24
25                  // Resetear la l gica del script
26                  carie.enabled = true;
27
28                  // Resetear el estado interno
29                  typeof(QuitarCaries)
30                      .GetField("yaFueEliminada", System.Reflection
```

```
                        .BindingFlags.NonPublic | System.Reflection
                        .BindingFlags.Instance)
                    ?.SetValue(carie, false);
            }
        }

        if (modeloExito != null)
            modeloExito.SetActive(false);

        Debug.Log("Reto reiniciado correctamente");
    }

    public void CarieEliminada()
    {
        contadorCaries++;

        Debug.Log("Caries eliminadas: " + contadorCaries);

        if (contadorCaries >= caries.Count)
        {
            MostrarVictoria();
        }
    }

    private void MostrarVictoria()
    {
        if (modeloExito != null)
            modeloExito.SetActive(true);

        if (uiManager != null)
            uiManager.MostrarVictoriaUI();
        else
            Debug.LogWarning("UIManager no asignado en
                ManagerRetoDental");
    }
}
```

Listing 3: Script ManagerRetoDental.cs

6.3. Modo Reto 2: Extraer Muelas

Este reto simula el procedimiento clínico de extracción dental mediante tres pasos secuenciales. Se activa desde el botón "Reto2 Muelas" en el menú de retos y utiliza el objeto Challenge2Object.

Pasos del reto:

Aplicar anestesia: el usuario debe tocar o arrastrar el objeto anestesia, lo que ejecuta el script AnestesiaPaso.cs y valida el inicio del procedimiento. Aflojar la muela: una vez anestesiada, el usuario utiliza las pinzas virtuales, activando el script AflojarPaso.cs, que simula el movimiento de aflojamiento. Extraer la muela: al completarse los pasos anteriores, el usuario puede arrastrar o tocar la muela para extraerla, validado por el script ExtraerPaso.cs. La lógica completa es gestionada por ExtraerMuelaManager.cs, que asegura que los pasos se ejecuten en el orden correcto. Al finalizar el proceso, se muestra un VictoriaCanvas con efectos visuales de éxito y se activa un nuevo trofeo.

7. Gestión de la Interfaz de Usuario (UI)

El script principal encargado de la navegación, visibilidad de objetos y control de flujo en los modos de juego es UIManager.cs.

Sus funciones principales incluyen:

- Navegar entre las distintas pantallas (MenúPrincipal, MenuAprende, MenuPrincipalRetos, MenuReto1, etc.).
- Activar o desactivar los objetos learnContent y challengeContent según el modo seleccionado.
- Ejecutar la función ResetearReto() al cambiar de reto o reiniciar una actividad.
- Mostrar los elementos de retroalimentación positiva como los Canvas de victoria.
- Controlar la visibilidad del ImageTarget en todo momento, evitando que desaparezca durante los cambios de escena. L
- os botones están conectados mediante onClick.AddListener() a funciones públicas del UIManager, como MostrarMenuPrincipal(),
- MostrarModoAprende(), MostrarMenuRetos(), IniciarReto1Caries(), entre otras.

```
using UnityEngine;

public class UIManager : MonoBehaviour
{
    [Header("Men s Principales")]
    public GameObject menuPrincipal;
    public GameObject menuAprende;
    public GameObject menuPrincipalRetos;
    public GameObject menuReto1Caries;
    public GameObject victoriaCanvas;

    [Header("Contenido del Image Target")]
    public GameObject learnContent;
    public GameObject challengeContent;

    [Header("Image Target")]
    public GameObject imageTarget;

    [Header("Referencias Clave")]
    public ManagerRetoDental managerRetoDental;

    void Start()
    {
        ConfigurarEstadosIniciales();
        MostrarMenuPrincipal();
    }

    void ConfigurarEstadosIniciales()
    {
        learnContent.SetActive(false);
        challengeContent.SetActive(false);
        victoriaCanvas.SetActive(false);
        imageTarget.SetActive(true);
    }

    public void MostrarMenuPrincipal()
    {
        menuPrincipal.SetActive(true);
        menuAprende.SetActive(false);
        menuPrincipalRetos.SetActive(false);
        menuReto1Caries.SetActive(false);
```

```
42          victoriaCanvas.SetActive(false);
43
44          OcultarTodoContenido();
45          if (managerRetoDental != null && managerRetoDental.
               modeloExito != null)
46              managerRetoDental.modeloExito.SetActive(false);
47      }
48
49      public void MostrarModoAprende()
50      {
51          menuPrincipal.SetActive(false);
52          menuAprende.SetActive(true);
53          learnContent.SetActive(true);
54          challengeContent.SetActive(false);
55          victoriaCanvas.SetActive(false);
56
57          if (managerRetoDental != null && managerRetoDental.
               modeloExito != null)
58              managerRetoDental.modeloExito.SetActive(false);
59      }
60
61      public void MostrarMenuRetos()
62      {
63          menuPrincipal.SetActive(false);
64          menuPrincipalRetos.SetActive(true);
65      }
66
67      public void IniciarReto1Caries()
68      {
69          menuPrincipalRetos.SetActive(false);
70          menuReto1Caries.SetActive(true);
71          challengeContent.SetActive(true);
72          managerRetoDental.ResetearReto();
73      }
74
75      public void VolverDesdeReto1()
76      {
77          menuReto1Caries.SetActive(false);
78          victoriaCanvas.SetActive(false);
79          challengeContent.SetActive(false);
80          MostrarMenuRetos();
```

```
81      }
82
83      public void MostrarVictoriaUI()
84      {
85          victoriaCanvas.SetActive(true);
86          menuReto1Caries.SetActive(false);
87      }
88
89      void OcultarTodoContenido()
90      {
91          learnContent.SetActive(false);
92          challengeContent.SetActive(false);
93      }
94  }
```

Listing 4: Script UIManager.cs

8. Sistema de Eventos

La lógica del proyecto Dental AR está altamente basada en eventos internos entre objetos:

- En el reto de caries, cada QuitarCaries.cs emite una señal al ManagerRetoDental para registrar su eliminación.
- En el reto de muelas, cada paso (AnestesiaPaso, AflojarPaso, ExtraerPaso) realiza validaciones con el ExtraerMuelaManager antes de permitir que el usuario continúe.
- Los objetos Trofeo están posicionados de forma externa a los retos para evitar ser ocultados accidentalmente por cambios de visibilidad en la jerarquía.
- Esta estructura basada en eventos mantiene el código modular, reutilizable y fácilmente escalable.

9. Elementos Visuales y Sprites

Para mantener una experiencia visual atractiva, especialmente orientada a público infantil, se han utilizado los siguientes recursos gráficos:

- Fondos adaptados a resolución nativa de dispositivos modernos (iPhone 13/14) para asegurar compatibilidad y calidad visual.
- Botones personalizados, con iconografía clara y textos descriptivos, alineados al estilo cartoon.

- Sprites diseñados con herramientas como Illustrator y Photoshop, complementados con elementos generados por DALL · E para crear instrucciones llamativas, amigables y temáticamente coherentes con el entorno odontológico.

10. Créditos y Colaboradores

- Diseño y desarrollo: Flores Cruz Daniel
- Sprites: Flores Cruz Daniel + ChatGpt