d. Prototipo

1. Proporcione un enlace a su prototipo de software (que debe crearse de acuerdo con las pautas anteriores). En la calificación, buscaremos ver qué tan bien ha traducido su framework de diseño en una representación digital. ¿Tus ideas se han traducido bien n la pantalla?
2. Escriba una descripción general de cómo funciona su prototipo, utilizando capturas de pantalla para ilustrar la funcionalidad y transmitir cómo el usuario puede navegar por a aplicación.
3. Describa los aspectos internos de su implementación: ¿qué lenguajes de programación, framework, etc. utilizó para desarrollar su prototipo? Analice las decisiones de diseño importantes que tomó en la implementación: ¿su elección de lenguajes de programación, por ejemplo, afectó sus decisiones de diseño? También analice cómo los problemas de implementación pueden haber afectado la usabilidad de su interfaz.

e. Cronograma de trabajo para la implementación de los requerimientos , el mismo que deberá estar planificado en 5 entregables

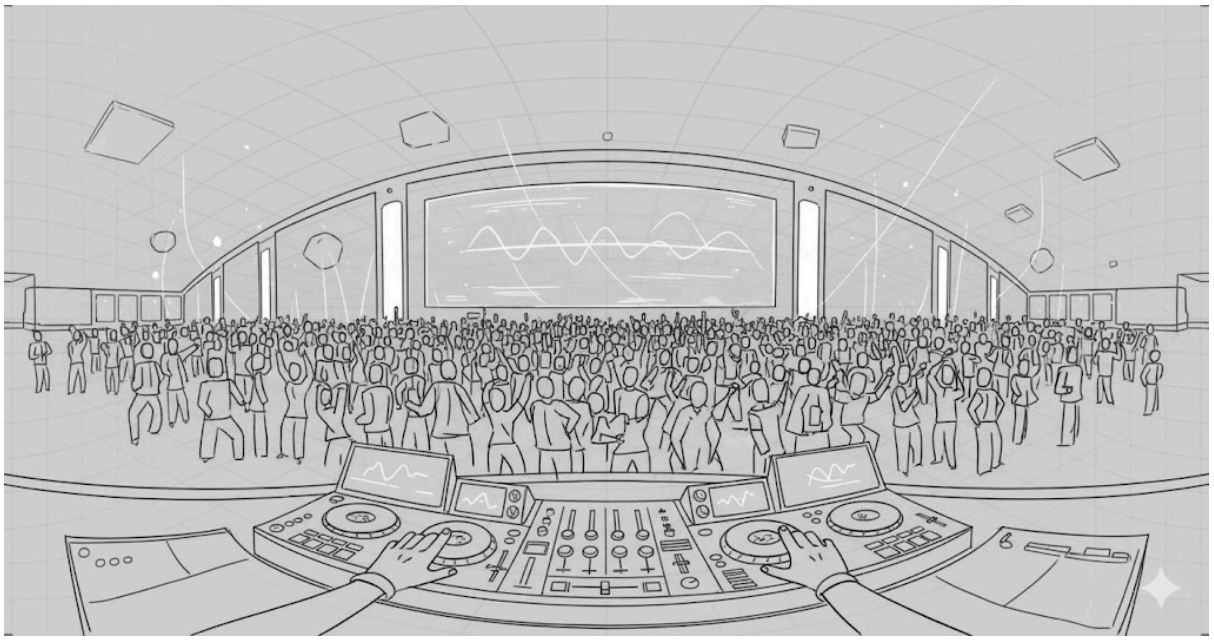
- Entregable 1 al 15% (Semana 11)
- Entregable 2 al 30% (Semana 12)
- Entregable 3 al 50%, (Semana 13)
- Entregable 4 al 70% (Semana 14)
- Entregable 5 al 100% (Semana 16)

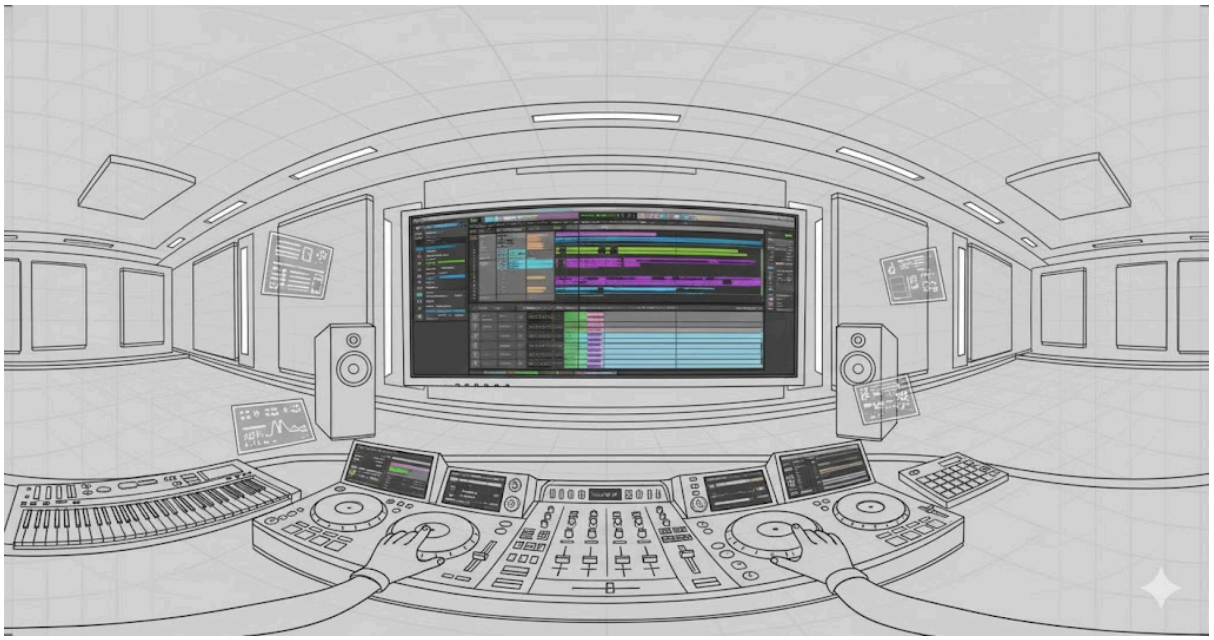
Descripción del prototipo

El objetivo principal del juego es que el usuario (el DJ) mezcle música en tiempo real para mantener a la audiencia virtual entretenida o para practicar en solitario.

1. **Controles y Mapeo 1:1:** El usuario usa controles de movimiento de VR. Sus manos virtuales, visibles en la imagen, imitan los movimientos de sus manos reales. La **retroalimentación háptica** (vibración) simula la sensación de tocar los discos (para el *scratching*) o sentir el "clic" de los botones y *faders*.
2. **Mecánicas de Mezcla:** El usuario interactúa directamente con el equipo en la cabina:
    - **Platos (Turntables):** Agarrar físicamente el disco virtual para hacer *scratch*.
    - **Sliders:** Ajustar el *pitch* (tono) y el *tempo* (velocidad)
    - **Dar "play/pausa" y “cue”**
    - **Mezcladora (Mixer):** Usar los *faders* de volumen para las canciones, el *crossfader* para cambiar entre platos, y las perillas de ecualización (EQ) para ajustar bajos, medios y agudos.
    - **Efectos (FX):** Activar y manipular perillas para añadir efectos como *reverb*, *delay* o filtros, tal como se haría en un equipo real.
3. **El Bucle del Juego (Game Loop):**
    - **Biblioteca de Música:** El usuario tendría una lista de canciones (virtual) para elegir o subir sus propios archivos.

- **Práctica en solitario:** El usuario tendrá la opción de practicar en un estudio en solitario, un ambiente sin presión para practicar todo lo que quiera.
- **Reacción del Público:** El usuario también podrá tocar frente a un público en una discoteca. La audiencia (visible al frente) reaccionaría dinámicamente. Si el DJ hace una buena transición, mantiene el ritmo y usa efectos de forma atractiva, el público bailará más, gritará y la energía del lugar subirá.
- **Retroalimentación:** Si el DJ comete errores (mala mezcla, silencio, transiciones fallidas), el público dejará de bailar, empezará a abuchear o incluso a irse.





Cómo navegaría el usuario (Navegación y UI)

1. Navegación en el Menú Principal (Fuera de la actuación)

Al iniciar el juego, el usuario estaría en el estudio.

- **Puntero Láser:** El usuario usaría sus controles como punteros láser para apuntar y seleccionar opciones en un menú 2D flotante.
- **Opciones:** Aquí seleccionaría modos de juego ("Discoteca" para pasar al escenario, "Sesión de práctica" para quedarse en el estudio), podría personalizar su equipo (elegir diferentes *decks* o mezcladoras) o accedería a tutoriales.

2. Navegación en el Juego

Mientras se está en el escenario, no se puede pausar para buscar en un menú. Durante la práctica en el estudio, se puede hacer de todo sin restricciones.

- **Pantallas de los Decks:** Las pantallas pequeñas integradas en los *decks* (visibles en la imagen) serían la interfaz principal. El usuario usaría perillas o *touchpads* en el *deck* virtual para navegar por su biblioteca de canciones.
- **Gestos de "Agarrar":** Para cargar una canción, el usuario la seleccionaría en la pantalla y luego usaría un gesto de "agarrar" (apretando el gatillo del control) para "arrastrarla" al plato virtual deseado.
- **Interfaz Holográfica/Muñeca:** Para salir de la discoteca, el usuario podría mirar su smartwatch virtual. Ahí, podría presionar un botón para salir y terminar el juego prematuramente.

Descripción de la implementación

1. Aspectos Internos: Lenguajes y Frameworks

- **Motor de Juego (Framework):** Unity.
- **Lenguaje de Programación:** C#.
- **SDK de VR:** Meta XR SDK.
- **Gestión de Audio:** Utilizamos el sistema de audio nativo de Unity (**AudioSource** y **AudioClip**) para la reproducción de los archivos de audio **Ogg Vorbis**. Se desarrollaron *scripts* personalizados en C# para manejar la sincronización, el *pitch shifting* (cambio de velocidad) y los efectos de *scratching*.

2. Decisiones de Diseño Impulsadas por la Implementación

Influencia de Unity y C# (Arquitectura Basada en Componentes):

La arquitectura de Unity, basada en GameObjects y Componentes, dictó nuestro diseño de software. En lugar de un sistema monolítico, diseñamos el equipo de DJ de forma modular.

- **Decisión de Diseño:** Cada elemento interactivo del *deck* (un *fader*, un botón, un plato de vinilo) es un GameObject independiente con sus propios *scripts* (componentes C#).
- **Ejemplo:** Un Fader.cs gestiona solo el movimiento en su eje y reporta su valor. Un Turntable.cs maneja la rotación y la lógica de *scratching*. Un AudioManager.cs central escucha los eventos de estos componentes para modificar el audio.

- **Ventaja:** Este diseño orientado a objetos permite crear un prototipo escalable y fácil de depurar. Añadir un nuevo botón no requería modificar el código del *fader*.

Influencia del Meta Quest 2 (Hardware Móvil):

El Quest 2 es una plataforma móvil, lo que significa que tiene recursos de CPU y GPU limitados.

- **Decisión de Diseño (Gráficos):** Se busca priorizar la tasa de fotogramas sobre la fidelidad visual para evitar el mareo. Esto significa:
    - Usar **modelos 3D *low-poly***..
    - Utilizar el **Universal Render Pipeline (URP)** de Unity, que está optimizado para rendimiento en dispositivos móviles.
- **Decisión de Diseño (Audio):** Optamos por el formato **Ogg Vorbis** en lugar de MP3. Aunque un MP3 de 320kbps ofrece alta calidad, su decodificación es computacionalmente costosa. Ogg Vorbis proporciona una calidad comparable (o superior) pero con un **costo de decodificación en CPU significativamente menor**, lo cual es vital en el Quest 2.
    - A pesar de esta eficiencia, para eliminar por completo el riesgo de picos de CPU, decidimos **precargar las pistas de audio en la memoria (RAM)** en lugar de transmitirlas (*streaming*). Esto consume más RAM, pero garantiza que la CPU esté totalmente libre para la física de interacción y el *tracking* durante la mezcla.

3. Problemas de Implementación y su Efecto en la Usabilidad

Todavía no tenemos problemas

Cronograma propuesto

Entregable 1 (15% - Semana 11)

Objetivo: Tener un entorno de prueba en el Quest 2 donde el usuario puede ver sus manos e interactuar con *un* objeto del equipo de DJ.

- **Proyecto y VR:**
    - Crear proyecto en Unity (URP - Universal Render Pipeline).
    - Configurar el Meta XR SDK.
    - Implementar el *rig* de VR: El usuario puede ver sus manos (controladores).
- **Modelos y Entorno Básico:**
    - Importar modelos 3D *placeholder* para la mezcladora y los platos.
    - Crear la "Escena de Prueba" para alinear la altura y escala del equipo de DJ.
- **Interacción Núcleo:**
    - Implementar la interacción física básica: El usuario puede "tocar" los *placeholders* del equipo.
    - Script para *un solo* componente: Un *fader* de volumen que se puede agarrar y mover, y que imprime su valor en la consola (ej. 0.8).

Resultado al 15%: El usuario está en una sala vacía, ve sus manos y puede mover un *fader* de volumen (RF004).

Entregable 2 (30% - Semana 12)

Objetivo: Implementar el sistema de audio. El usuario debe poder mezclar dos canciones usando el equipo básico.

- **Sistema de Carga de Audio:**
  - Implementar la lógica de carga dinámica de Ogg Vorbis.
  - Crear una UI temporal (menú 2D simple) para seleccionar 2 canciones de una lista de 10.
  - Asignar las canciones a los AudioSources de "Plato A" y "Plato B".
- **Mezcla Básica:**
  - Conectar el *fader* de volumen del Entregable 1 al AudioSource.volume.
  - Implementar el *Crossfader*: Un *script* que controla el volumen de ambos platos simultáneamente (sube A mientras baja B).

Resultado al 30%: El usuario puede cargar dos canciones y mezclarlas usando los *faders* de volumen y el *crossfader* (**RF004**, **RF002**). La mecánica de audio principal está probada.

Entregable 3 (50% - Semana 3)

Objetivo: Completar el primer ambiente y la funcionalidad completa del *deck*. El prototipo debe ser *jugable* en modo solitario.

- **Ambiente 1: Estudio:**
  - Importar los assets 3D finales para el estudio en solitario.
  - Implementar la iluminación.
- **Interacción Avanzada:**
  - Implementar el resto de interacciones del *deck*:
    - **Platos (Turntables):** Lógica de *Scratching* (vincular la velocidad al AudioSource.pitch y time).
    - **Perillas (Knobs):** Scripts para las perillas de ecualización (EQ: Bajos, Medios, Agudos).
    - **Botones:** Botones de "Play/Pausa" y "Cue".

Resultado al 50%: Un prototipo funcional. El usuario está en un estudio detallado y tiene control total sobre el equipo de DJ (mezcla, EQs, scratching) (**RF003**, **RF005**, **RF009**).

Entregable 4: (70% - Semana 14)

Objetivo: Implementar el segundo ambiente y la audiencia.

- **Ambiente 2: Discoteca:**
  - Modelar e importar los assets 3D para la **Discoteca.**
- **Sistema de Audiencia:**
  - Crear o importar modelos 3D *low-poly* para el público (siluetas o modelos simples tipo *Mii*).

    - Implementar un *script* de audiencia básico: El público tiene dos animaciones (Idle y Bailando).
    - Lógica simple: Si el AudioSource principal está sonando, el público "Baila". Si no, Idle. (**RF008**)

Resultado al 70%: El usuario puede elegir entre el Estudio y la Discoteca. La discoteca tiene un público que reacciona a la música.

Entregable 5 (100% - Semana 16)

Objetivo: Presentar un prototipo estable, pulido y completo.

- **Pulido Avanzado (15%)**
    - Mejorar el sistema de audiencia (ej. que reaccionen a la mezcla o al *crossfader*).
    - Añadir un *feature* de audio extra (ej. un filtro "low/high pass" en la mezcladora). (**RF003**)
    - Crear un menú principal sólido para seleccionar el ambiente.
- **Bug Fixing (Arreglo de errores):**
    - Corregir errores con las manos.
    - Corregir problemas de carga de audio.
    - Ajustar la iluminación, reducir polígonos y optimizar *scripts* que consuman mucha CPU.
- **Usabilidad**
    - Ajustar la "sensibilidad" de las perillas y el *scratching*.
    - Implementar retroalimentación háptica (vibración) en los controles al tocar botones o hacer *scratch*.

Resultado al 100%: Un prototipo completo, optimizado y pulido.