Universidad Nacional Autónoma de México
Facultad de Ingeniería
Realidad Virtual y Aumentada
Manual Técnico – Huerto Urbano en Realidad Aumentada
Ramírez Andrés Daniela
Num Cuenta: 317121849
17 de noviembre de 2024

Índice

1. Blender

Utilicé Blender como herramienta principal para el modelado 3D de nuestra aplicación, dado que tengo un conocimiento más sólido en su uso. Esta plataforma me permitió crear modelos detallados y precisos para cada etapa de la planta de jitomate, optimizando el flujo de trabajo y logrando resultados que se alinean con los objetivos de la aplicación.

1.1. Creación de Modelos Para la aplicación AR Huertos Urbanos

Semana del 28 de Octubre al 4 de Noviembre

Durante estas semanas, llevé a cabo el desarrollo completo de cinco modelos que ilustran las etapas de crecimiento de una planta de jitomate, los cuales presenté en la aplicación final. Cada modelo fue creado desde cero en Blender, donde me encargué de todos los detalles, desde las macetas hasta las hojas y tallos de las plantas. En cada fase de modelado, trabajé para capturar con precisión los cambios en la apariencia de la planta a medida que pasa por sus diferentes etapas de desarrollo.

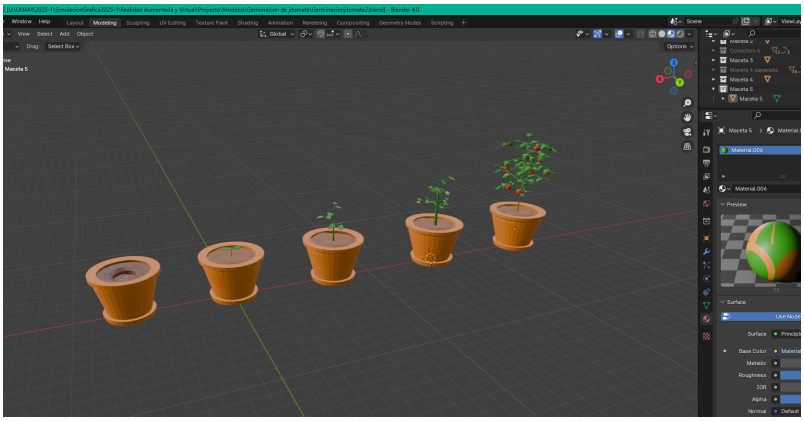


Figura 1: Modelos: Etapas de Jitomate

2. Unity and GitHub

2.1. Creación de un Repositorio

Semana del 4 de Noviembre al 11 de Noviembre

El primer paso en el desarrollo fue crear un repositorio en GitHub, lo cual es fundamental para implementar un control de versiones adecuado. Esto permitió gestionar mejor el progreso y las versiones del trabajo.

Primero, creé el repositorio en GitHub, configurando parámetros como el nombre del proyecto y el archivo .gitignore, adaptado a Unity para ignorar archivos innecesarios. Luego, cloné el repositorio en mi máquina local para poder trabajar en Unity con un enlace directo a este repositorio. Esto asegura que cada cambio realizado en Unity pueda ser rastreado, revertido o actualizado en GitHub, manteniendo un control preciso de las versiones.

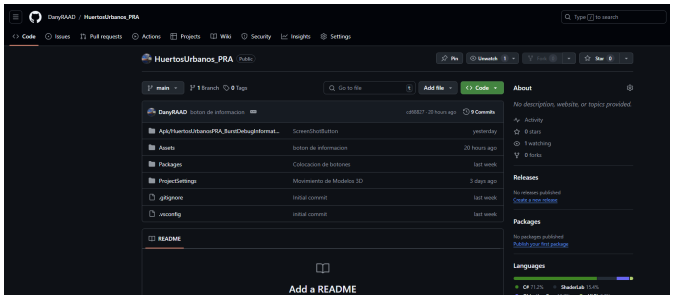


Figura 2: Repositorio en GitHub

2.2. Creación de Proyecto en Unity enlazado a GitHub

A continuación, se creó un proyecto vacío en Unity configurado para 3D con el Built-in Render Pipeline. Es importante guardarlo en una ubicación conocida, ya que luego se moverían estos archivos al repositorio previamente clonado.

Una vez creado el proyecto, se accedió a la carpeta donde Unity guardó el proyecto, se seleccionaron todos los archivos y se copiaron a la carpeta del repositorio. Con los archivos en la ubicación correcta, se abrió el proyecto desde allí. Para hacerlo, en Unity Hub se seleccionó Add y luego Add project from disk, donde se buscó y seleccionó la carpeta del repositorio.

Ahora el proyecto está enlazado a GitHub y, con la aplicación de GitHub Desktop, se pueden realizar commits y push automáticamente, facilitando el control de versiones y mejorando la organización del proyecto.

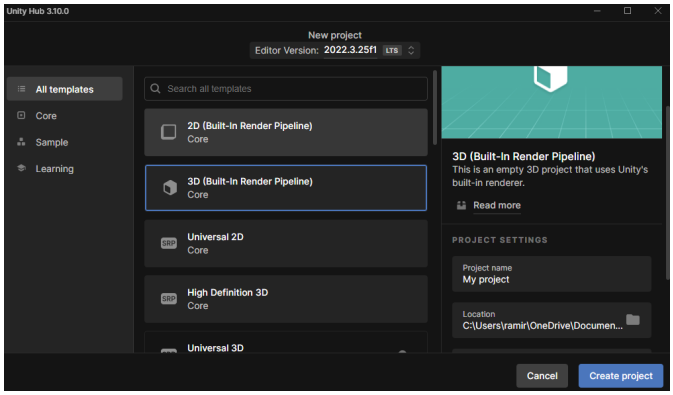


Figura 3: Creación de Proyecto en Unity

2.3. Configuración del Proyecto

2.3.1. Módulo de Android en versión de Unity

Para desarrollar para Android, es necesario tener instalado el módulo de Android en Unity. Este módulo incluye el Android SDK, NDK y OpenJDK, necesarios para compilar y probar la aplicación en dispositivos Android.

Para instalarlo, abrí Unity Hub, fui a la sección de Installs, seleccioné la versión de Unity y luego hice clic en Add modules. Ahí seleccioné el módulo de Android. Con estos elementos configurados, pude optimizar el proyecto para dispositivos Android directamente desde Unity.

Figura 4: Módulo de Android

2.3.2. Cambio de plataforma dentro de Build Settings

Una vez dentro del proyecto, hice algunos ajustes. Primero, cambié la plataforma de desarrollo a Android. Para hacerlo, fui a la pestaña File y seleccioné Build Settings. En el panel derecho, elegí Android y luego hice clic en el botón Switch Platform. Esto permitió que Unity optimizara el proyecto para Android y activara las opciones de configuración específicas para esta plataforma.

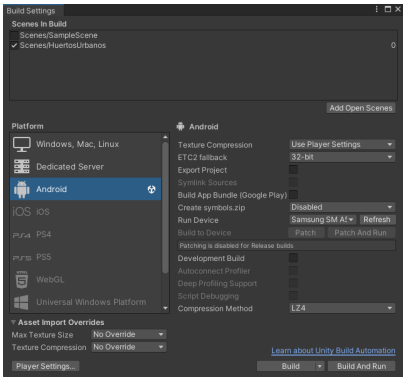


Figura 5: Cambio a Android en Build Settings

2.3.3. Input System

Como el proyecto usa un sistema de entrada táctil, es necesario configurar el nuevo sistema de entrada (Input System Package) para evitar problemas de compatibilidad. A partir de versiones recientes, Unity maneja dos tipos de sistemas de entrada: el sistema antiguo y el nuevo.

Para configurarlo, fui a Edit >Project Settings >Player >Other Settings y, en Active Input Handling, seleccioné Input System Package (New). Esto provocó que Unity solicitara un reinicio para cargar correctamente los cambios.

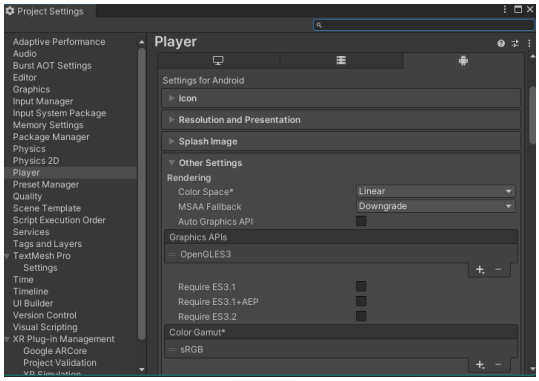


Figura 6: Input System New

2.3.4. Packages Para Realidad Aumentada

Para la implementación de la Realidad Aumentada, instalé algunos paquetes específicos en lugar de usar Vuforia. Esto optimizó el manejo de la RA en el proyecto.

Accedí al `Package Manager` desde `Window >Package Manager`, seleccioné `Unity Registry` y busqué los siguientes paquetes:

- **AR Foundation**: Este paquete es esencial para la interacción con el entorno físico en dispositivos móviles.
- **Google ARCore XR Plugin**: Este *asset* es necesario para integrar la plataforma ARCore de Google en la aplicación, aprovechando las capacidades de RA específicas de Android.

Al instalar estos dos paquetes, pude manejar la Realidad Aumentada en la aplicación sin necesidad de Vuforia, garantizando compatibilidad con múltiples dispositivos y mejor control de las funcionalidades de RA.

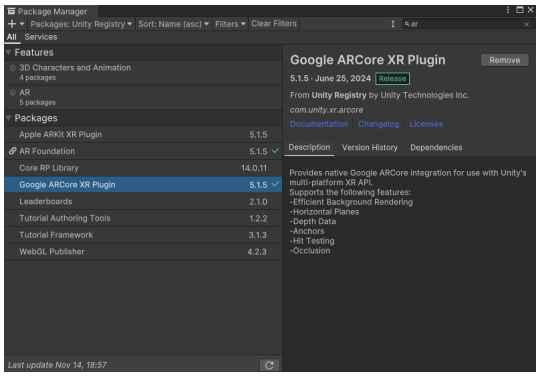


Figura 7: Packages para AR

2.3.5. Assets necesarios para la implementación

Para la funcionalidad del proyecto, incorporé algunos **assets** de la `Asset Store`. El primero de ellos es `DOTween`, que facilita la animación de los botones de la interfaz. El segundo es `Native Gallery for Android & iOS`, que permite capturar `screenshots` mediante un botón.

Para agregarlos en Unity, accedí a la `Asset Store`, busqué ambos **assets**, y los añadí al proyecto. Luego, en la pestaña `Package Manager`, los descargué e importé para tenerlos listos para su uso.

2.3.6. Configuración de XRCore

Finalmente, configuré `XRCore` para asegurarme de que la aplicación funcionara correctamente con la Realidad Aumentada en dispositivos Android. Para hacerlo, fui a `Edit >Project Settings >XR Plug-in Management` y verifiqué que `Google ARCore` estuviera seleccionado en la pestaña de `Android`.

Con esta configuración, habilité las funcionalidades necesarias de Realidad Aumentada específicas para Android, asegurando que el proyecto estuviera listo para aprovechar las capacidades de `ARCore`.

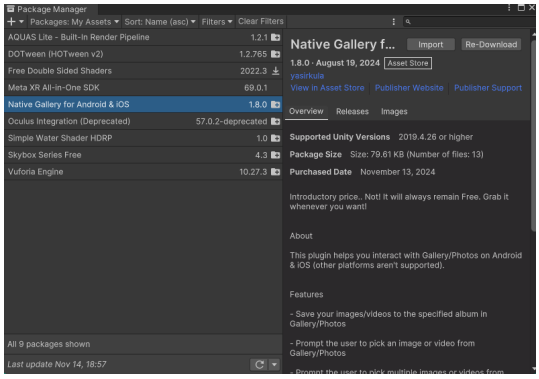


Figura 8: Assets

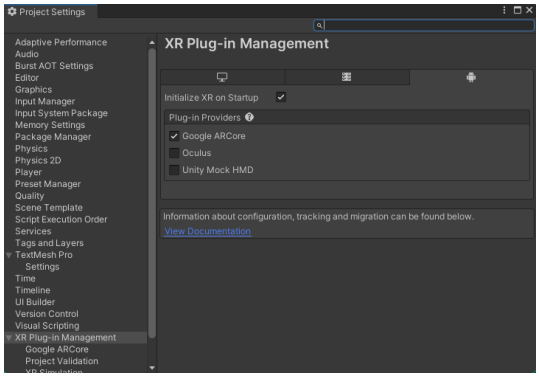


Figura 9: Configuración de XRCore

3. Desarrollo de la App dentro de Unity

3.1. Creación de la interfaz

Para comenzar con el desarrollo de la interfaz de nuestra aplicación, me aseguré de crear una estructura visual clara e intuitiva que facilitará la experiencia de usuario. Comencé creando un objeto vacío en Unity que contendrá tres `Canvas`, los cuales servirán para cambiar entre tres tipos de pantallas con diferentes interfaces.

Para cada `Canvas`, diseñé una interfaz específica, junto con sus respectivos botones a los cuales asigné íconos representativos como imágenes. Esto permite identificar cada funcionalidad de forma visual y accesible. Los tres `Canvas` fueron nombrados de la siguiente manera:

- **MainMenu**: Esta es la pantalla principal desde la cual se accede a las opciones generales de la aplicación.
- **SelectItems**: Aquí el usuario puede seleccionar los elementos que desea visualizar o interactuar en la aplicación.
- **ARPosition**: En esta pantalla, se encuentra la interfaz de Realidad Aumentada (RA), donde se posicionarán los objetos en el entorno físico.

Esta estructura modular me permite alternar entre diferentes pantallas dentro de la aplicación de manera organizada, asegurando que la interfaz sea intuitiva y esté bien adaptada a las funciones que ofrece.

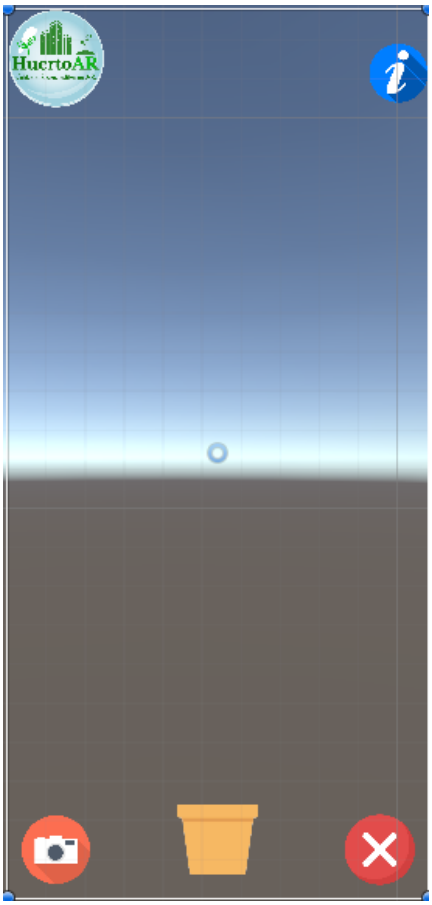


Figura 10: MainMenu

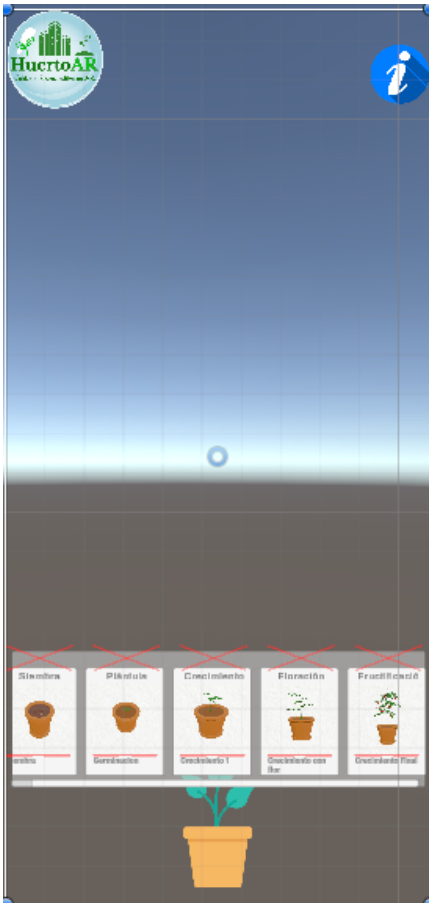


Figura 11: SelectItems

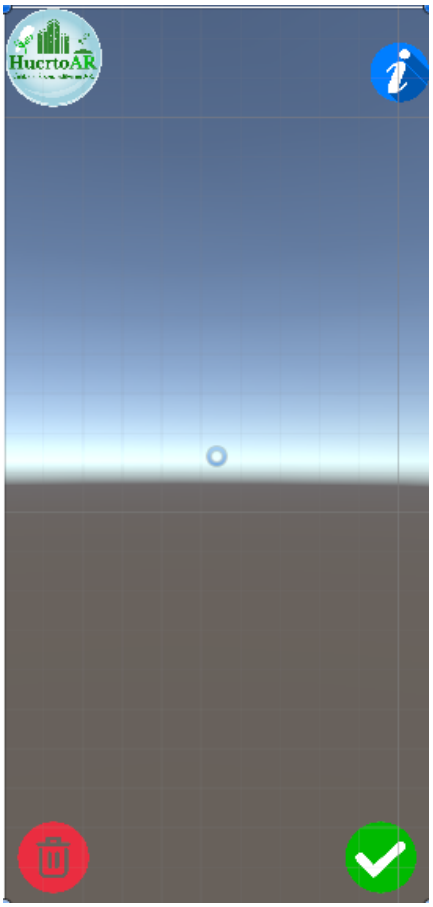


Figura 12: ARPosition

Para la navegación de la interfaz, creé un código que asigné al `UI Manager`, el cual permite desplazarnos dentro de la aplicación y pasar de una pantalla a otra. Una vez asignado el código, pude agregar los métodos correspondientes en los eventos `OnClick` de cada botón de la interfaz para activar la navegación entre las diferentes pantallas.

3.2. Detección de Puntos

Para que mi aplicación funcione con realidad aumentada, asigné un componente que permite detectar superficies, lo cual es esencial para colocar nuestros modelos en un entorno real. Para lograr que detecte las superficies, comencé creando un sistema de partículas. Luego, añadí una cámara `AR Session` y un objeto `AR Origin`. Este último contiene la cámara que facilita la interacción con la realidad aumentada.

A `AR Origin` le añadí varios componentes, incluyendo el `AR Point Cloud Manager`. Creé un prefab utilizando el sistema de partículas previamente configurado y lo asigné al `AR Point Cloud Manager`. De esta forma, la aplicación es capaz de detectar planos en el entorno, lo cual resulta en una experiencia de realidad aumentada interactiva y funcional.

3.3. Integración de Modelos 3D en ScrollView

Para evitar crear cada botón manualmente, opté por crear un `prefab` del botón en el `Content` de nuestro `ScrollView`. Configuré este botón para que incluya un pequeño título, una imagen y una des-

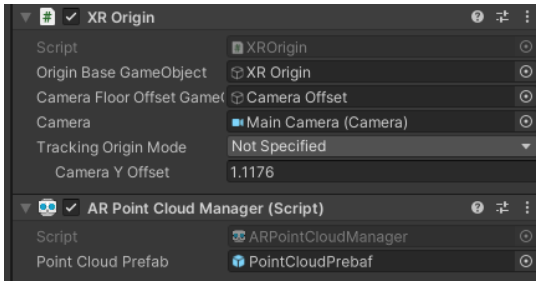


Figura 13: Detección de puntos

cripción de nuestros modelos. Para gestionar la lógica detrás de estos elementos, asigné un script llamado `ItemButtonManager` y otro script para definir cada `Item`.

El script de `Item` facilita la creación de un elemento para cada modelo desde la interfaz de Unity. Al ir a la opción `Create` en Unity, aparece la opción de `Item`, y al seleccionarla, se puede añadir un título, una imagen, una descripción y el prefab de cada botón correspondiente a los modelos creados previamente. De este modo, cada vez que generemos un nuevo `Item` y modelo, se creará automáticamente un botón que representa cada uno.

Figura 14: itemButton

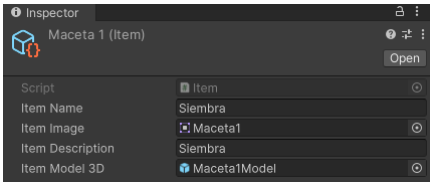


Figura 15: Item

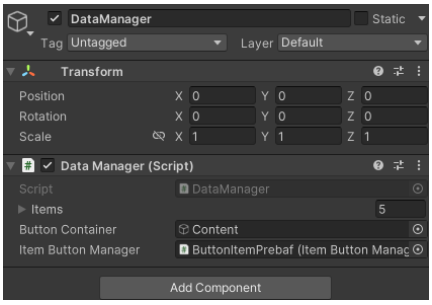


Figura 16: datamanager

Para permitir la interacción directa con el modelo en RA, modifiqué el script de `ItemButtonManager` e implementé un nuevo script llamado `ARInteractionManager`. Este último gestiona la interacción con el modelo 3D en el entorno de realidad aumentada. Para asegurar que el sistema táctil funcionara de manera óptima y evitar conflictos, programé ambos scripts utilizando el *New Input System* de Unity.

4. Función de ScreenShot

Para implementar la función de captura de pantalla, utilicé la librería *Native Gallery*, que previamente había instalado. Creé un script específico para realizar esta función y asigné dicho script al botón de captura de pantalla configurado en pasos anteriores.

Adicionalmente, para proporcionar retroalimentación al usuario, desarrollé un sistema que muestra una leyenda indicando que la captura fue guardada en el dispositivo. Para lograrlo, creé un cuadro de texto en el *Canvas* y diseñé un script encargado de mostrar el mensaje de confirmación al presionar el botón.

Finalmente, configuré un objeto vacío en la jerarquía, al cual asigné el script, y vinculé el cuadro de texto correspondiente arrastrándolo al campo adecuado. Con estos pasos completados, el botón de captura de pantalla quedó funcional y con la retroalimentación visual configurada.

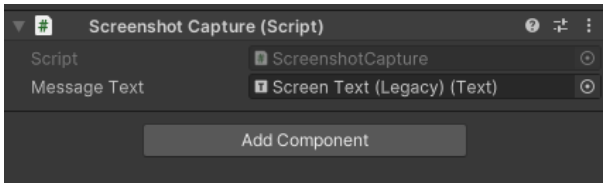


Figura 17: Screen Text

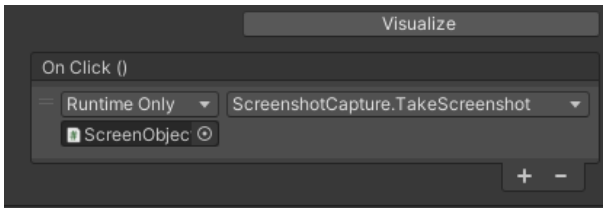


Figura 18: Screen Button

5. Botón de información

Continuando con el desarrollo de mi proyecto, decidí añadir una descripción detallada de cada modelo de la planta de jitomate y sus etapas de crecimiento. Esto responde al objetivo principal de mi aplicación: educar a las personas sobre cómo sembrar y cuidar sus propias plantas.

Para implementar esta funcionalidad, realicé primero una breve investigación sobre los cuidados necesarios para los jitomates, como la cantidad de riego, luz solar y humedad ideales. Una vez comprendida esta información, diseñé un panel informativo que incluye imágenes, títulos y descripciones organizadas de manera clara y visualmente atractiva.

Posteriormente, desarrollé un script que permite abrir y ocultar dicho panel cada vez que se presiona el botón de información. Asigné este script a un objeto vacío en la jerarquía y vinculé tanto el panel como el botón a través de los campos correspondientes en el inspector. Con estos pasos completados, logré que el panel informativo se muestre dinámicamente al interactuar con el botón, ofreciendo una experiencia educativa interactiva y práctica.

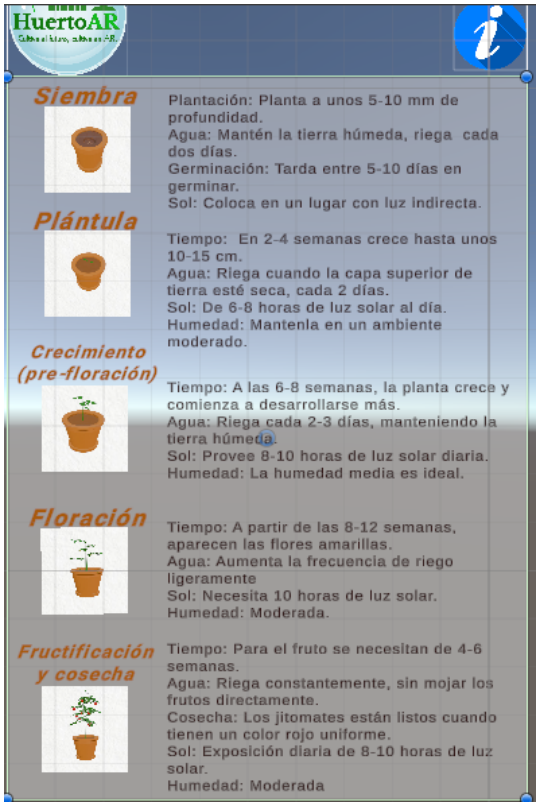
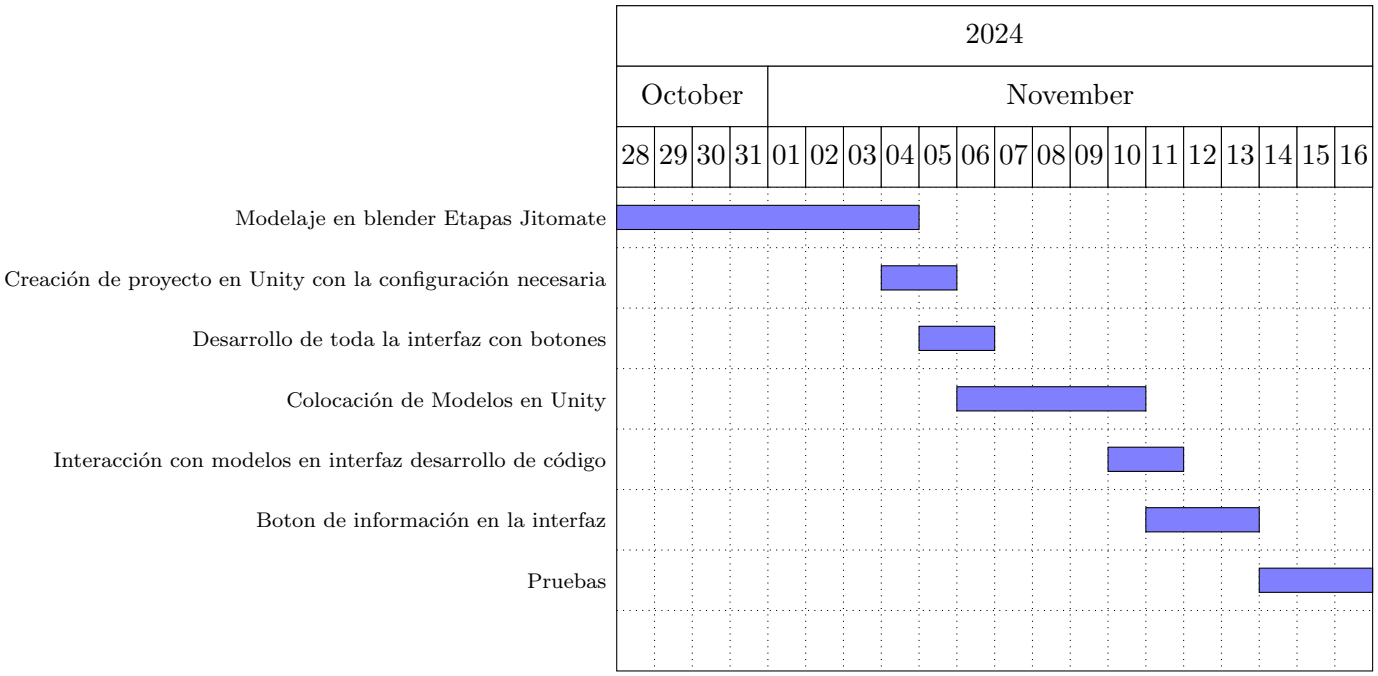


Figura 19: Panel

6. Cronograma de actividades realizadas

6.1. Diagrama de Gantt



Referencias

1. TurboSquid. (n.d.). *Tomato plant model*. Recuperado de https://www.turbosquid.com/3d-models/tomato-3ds-free/758044

2. Flaticon. (n.d.). *Tache icon*. Recuperado de https://cdn-icons-png.flaticon.com/512/753/753345.png

3. DesignBolts. (n.d.). *DSLR Camera icon*. Recuperado de https://icons.iconarchive.com/icons/designbolts/free-multimedia/512/Dslr-Camera-icon.png

4. Flaticon. (n.d.). *Plant pot icon*. Recuperado de https://cdn-icons-png.flaticon.com/512/3616/3616944.png

5. Freepik. (n.d.). *Delete icon*. Recuperado de https://cdn-icons-png.freepik.com/512/3807/3807871.png

6. Flaticon. (n.d.). *Check icon*. Recuperado de https://cdn-icons-png.flaticon.com/512/5610/5610944.png

7. Pixabay. (2016). *Info icon*. Recuperado de https://cdn.pixabay.com/photo/2016/06/15/15/02/info-1459077_640.png

8. Unity Adventure. (2021, 22 de noviembre). *¿Cómo crear una aplicación de Realidad Aumentada?* [Vídeo]. YouTube. Recuperado de https://www.youtube.com/watch?v=TI599JorZ5M