Guía de Trabajo y Colaboración del Proyecto VR: Casa Virtual

Fecha: Jueves, 10 de julio de 2025
Ubicación: Medellín, Antioquia, Colombia
¡Bienvenidos al equipo comprometido del proyecto de modelado 3D para Realidad Virtual! Nuestro objetivo es crear una experiencia inmersiva donde los usuarios puedan explorar una casa virtual. El éxito de este proyecto depende del compromiso de cada uno y de una colaboración fluida.

1. Cronograma de Desarrollo y Estudio Propuesto (Julio - Septiembre)

Este cronograma es nuestra hoja de ruta detallada. Exige un esfuerzo intensivo y constante para cumplir con la fecha de entrega final a finales de septiembre. ¡Cada semana cuenta!

Fase 1: Preparación y Profundización del Motor (Julio)

- **Semana del 7 al 13 de julio:**
  - **Reunión Clave (Jueves 10 de julio):** Definiremos roles definitivos y asignaremos las tareas iniciales. ¡Tu participación es crucial!
  - **Decisión y Configuración del Motor:** Formalizaremos la elección de Unreal Engine. Procederemos con la instalación y configuración inicial del proyecto.
  - **Estudio Profundo de Unreal Engine:** Los asignados a "Configuración del motor" y "Programación de interacciones" deben sumergirse en la documentación y tutoriales específicos de VR de Unreal Engine.
  - **Estudio profundo de GIT:** Los modeladores y los desarrolladores comenzarán con el estudio de GIT fundamental para el desarrollo del proyecto.
- **Semana del 14 al 20 de julio:**
  - **Modelado Arquitectónico Avanzado:** Empezaremos con la estructura principal de la casa y tratar de en esta semana hacerlo lo mas detallado posible.
  - **Integración Inicial de Modelos:** Los primeros modelos básicos se importarán a Unreal Engine. Haremos pruebas de exportación/importación y escalado.
  - **Flujo de Trabajo Git/GitHub:** Estableceremos un flujo de trabajo claro en el repositorio con ramas de desarrollo y prácticas de commit/push.
- **Semana del 21 al 27 de julio:**
  - **Modelado de Detalles Arquitectónicos:** Creación de puertas, ventanas detalladas, escaleras y columnas.
  - **Texturizado PBR (Physically Based Rendering):** Aplicación de texturas

realistas y optimizadas a los modelos.
    - **Unreal Engine - Materiales e Iluminación:** Configuración de materiales PBR e inicio de la iluminación básica en el motor para establecer la atmósfera.
- **Semana del 28 de julio al 3 de agosto:**
    - **Optimización de Activos:** Revisión y optimización de modelos y texturas existentes para asegurar un buen rendimiento en VR (reducción de polígonos, atlas de texturas).
    - **Unreal Engine - Navegación VR:** Implementación de un sistema básico de movimiento en primera persona adaptado para VR.

Fase 2: Desarrollo y Prototipado Interactivo (Agosto)

- **Semana del 4 al 10 de agosto:**
    - **Modelado de Mobiliario Principal:** Inicio del modelado de muebles clave para las primeras habitaciones.
    - **Interacciones Básicas VR:** Programación de interacciones simples (abrir/cerrar puertas, encender/apagar luces).
    - **Documentación de Procesos:** Comenzaremos a documentar decisiones de diseño, convenciones de nomenclatura y estilo.
- **Semana del 11 al 17 de agosto:**
    - **Continuación del Modelado:** Creación de más mobiliario y objetos decorativos.
    - **Optimización Constante:** Seguimos optimizando todos los activos nuevos.
    - **Unreal Engine - Iluminación Avanzada:** Refinamiento de la iluminación (luces baked, dinámicas), sombras y efectos de post-procesado para mejorar la calidad visual y la inmersión.
- **Semana del 18 al 24 de agosto:**
    - **Integración Masiva de Activos:** Importación e integración de la mayor parte del mobiliario y objetos en el entorno VR.
    - **Pruebas de Rendimiento VR:** Pruebas constantes en el motor y en hardware VR para identificar y solucionar cuellos de botella.
    - **Refinamiento de Interacciones:** Depuración y mejora de las interacciones programadas.
- **Semana del 25 al 31 de agosto:**
    - **Modelado de Exteriores:** Inicio del modelado de elementos exteriores (terreno, vegetación, etc.).
    - **Integración de Audio:** Implementación de sonidos ambientales y efectos de sonido.
    - **Documentación de Avances:** Actualización detallada de la documentación.

Fase 3: Refinamiento, Pulido y Preparación para la Entrega (Septiembre)

- **Semana del 1 al 7 de septiembre:**
  - **Detalle y Acabado Final:** Añadir detalles finos y pulir superficies.
  - **Optimización Avanzada:** Aplicar técnicas avanzadas (occlusion culling, LODs, simplificación de mallas).
  - **Pruebas de Usuario Internas:** El equipo realizará pruebas exhaustivas para identificar errores y mejoras.
- **Semana del 8 al 14 de septiembre:**
  - **Pulido de Interacciones:** Refinamiento final de las interacciones para una experiencia VR intuitiva.
  - **Interfaz de Usuario (UI) en VR:** Si es necesario, diseño e implementación de UI adaptados a VR.
  - **Corrección de Errores Críticos:** Enfoque en solucionar los bugs más importantes.
- **Semana del 15 al 21 de septiembre:**
  - **Revisión General:** Revisión completa de todos los aspectos del proyecto (modelos, texturas, código, iluminación, rendimiento).
  - **Materiales de Presentación:** Creación de capturas de pantalla y videos para la entrega.
  - **Documentación Final:** Compilación y finalización de toda la documentación técnica y de diseño.
- **Semana del 22 al 28 de septiembre (Última Semana):**
  - **Pruebas de Entrega Finales:** Últimas pruebas exhaustivas para asegurar estabilidad y calidad.
  - **Ajustes Finales:** Pequeños retoques visuales, de rendimiento o interacción.
  - **Empaquetado para Entrega:** Compilación y empaquetado del proyecto final para la presentación.

2. Información sobre el Motor Gráfico: Unreal Engine

Hemos decidido usar **Unreal Engine** como nuestro motor principal para la experiencia VR. Es una elección estratégica por sus potentes capacidades y su liderazgo en gráficos y rendimiento para Realidad Virtual.

Ventajas clave de Unreal Engine para nuestro proyecto:

- **Gráficos de Alta Fidelidad:** Unreal Engine es un referente en la industria por su capacidad de crear visualizaciones fotorrealistas. Esto es ideal para una visualización arquitectónica de una casa.
- **Rendimiento VR Superior:** Ofrece un excelente soporte y optimización específica para VR, lo que es crucial para evitar el mareo y proporcionar una

experiencia fluida.
- **Blueprint (Scripting Visual):** Aunque también usa C++, su sistema de Blueprint permite programar interacciones y lógica compleja de forma visual, lo que puede acelerar el desarrollo y hacer que la lógica sea más accesible para aquellos con menos experiencia en programación.
- **Iluminación Avanzada (Lumen) y Geometría (Nanite):** Las tecnologías como Lumen (iluminación global en tiempo real) y Nanite (micropolígonos) en Unreal Engine 5 permiten un nivel de detalle visual y realismo ambiental excepcional.

Consideraciones:

- **Curva de Aprendizaje:** Unreal Engine tiene una curva de aprendizaje más alta que Godot o Unity debido a su complejidad y potencia. Sin embargo, el esfuerzo se justifica por los resultados.
- **Requisitos del Sistema:** Requiere hardware potente para un desarrollo eficiente.

3. Información Completa sobre Git y GitHub

Git es un sistema de control de versiones distribuido que nos permite rastrear cada cambio en nuestros archivos y coordinar el trabajo entre todos los miembros del equipo. **GitHub** es la plataforma en línea que alojará nuestro repositorio Git, facilitando esta colaboración.

¿Por qué es absolutamente crucial usar Git/GitHub en este proyecto?

1. **Historial Completo de Cambios: Tu "Máquina del Tiempo"**
    - Cada modificación que hagas en cualquier archivo (modelos 3D, texturas, código del motor) se registra con quién lo hizo, cuándo y por qué.
    - Esto es invaluable para:
        - **Depuración:** Si algo se rompe, podemos identificar rápidamente cuándo y por qué sucedió.
        - **Reversión:** Volver a una versión anterior del proyecto si algo sale mal o si un cambio no funciona como se esperaba.
        - **Entender la Evolución:** Ver cómo el proyecto ha crecido y cambiado con el tiempo.
2. **Colaboración Confiable y Simultánea**
    - Permite que múltiples personas trabajen en el mismo proyecto al mismo tiempo sin sobrescribir el trabajo de los demás.
    - Cada uno trabaja en su copia local del proyecto y luego "sube" (push) sus cambios al repositorio central.
3. **Manejo de Conflictos y Prevención de Pérdidas**
    - Si dos personas modifican la misma parte de un archivo, Git nos avisará del

"conflicto" y nos proporcionará herramientas para resolverlo, evitando la pérdida de trabajo. Una buena organización de tareas minimiza estos conflictos.

4. **Ramas de Desarrollo (Branches): Experimentación Segura**
    - Las ramas nos permiten aislar el trabajo en nuevas funcionalidades o correcciones de errores en "ramas" separadas.
    - Esto significa que puedes experimentar o desarrollar algo grande sin afectar la versión principal y estable del proyecto hasta que esté listo, probado y aprobado.
5. **Estándar de la Industria: Habilidad Profesional Indispensable**
    - Git y GitHub son el estándar de oro en el desarrollo de software y juegos a nivel mundial. Aprender a usarlos eficientemente no solo es vital para este proyecto, sino que es una habilidad profesional muy demandada en toda la industria tecnológica.

¿Por qué es OBLIGATORIO que todos comprendan y usen el software rápidamente?

Dado nuestro ajustado cronograma hasta finales de septiembre, la capacidad de usar Git/GitHub con soltura es fundamental y no negociable:

- **Evitar Pérdidas de Trabajo:** Reduciremos drásticamente el tiempo perdido lidiando con sobrescrituras accidentales o archivos perdidos.
- **Acelerar la Integración:** Un flujo de trabajo Git eficiente significa que los modelos 3D y las texturas de los modeladores llegan rápidamente a los desarrolladores del motor, y las actualizaciones del motor llegan a todos sin fricción.
- **Facilitar la Depuración:** Si surge un problema, el historial de Git nos permite identificar rápidamente cuándo y por qué sucedió, facilitando la corrección.
- **Aumentar la Productividad:** Al reducir la fricción en la colaboración y los problemas técnicos, el equipo puede dedicar más tiempo a las tareas creativas y técnicas reales del proyecto.
- **Manejo de Archivos Grandes (Git LFS):** Para los modelos 3D, texturas y archivos de audio, utilizaremos Git Large File Storage (Git LFS). Esto es esencial porque Git no está diseñado para manejar archivos binarios grandes directamente. Git LFS almacena una pequeña referencia a estos archivos en el repositorio de Git, mientras que el archivo real se guarda en un servidor separado. Esto mantiene nuestro repositorio ligero y rápido. **Es fundamental configurarlo para todos los tipos de archivos grandes desde el inicio.**

4. Composición de los Equipos (11 Personas)

Para optimizar nuestro flujo de trabajo y aprovechar las habilidades de cada uno, esta es la distribución de roles:

- **1x Líder/Coordinador del Proyecto (Pava):**
  - **Responsabilidades:** Gestión general del proyecto, comunicación con los profesores, asignación de tareas, resolución de conflictos y mantenimiento de la visión del proyecto. También apoyará activamente a los otros grupos.
- **4x Modeladores 3D:**
  - **Responsabilidades:** Creación de todos los activos 3D (arquitectura, mobiliario, props), texturizado y optimización rigurosa para VR.
- **5x Desarrolladores/Integradores de Motor:**
  - **Responsabilidades:** Configuración de Unreal Engine, programación de interacciones, integración de modelos, configuración de iluminación, optimización del rendimiento en VR y pruebas técnicas.
- **1x Documentación (Sandra):**
  - **Responsabilidades:** Mantener la documentación del proyecto, asegurar la consistencia en convenciones (nomenclatura, estilo), y problemas de calidad.

5. Cómo se Va a Trabajar en el Equipo de los Modeladores

Los modeladores trabajarán de forma **modular y coordinada**, aprovechando al máximo Blender y Git/GitHub:

- **División Modular de la Casa:**
  - La casa se dividirá en componentes pequeños y manejables para permitir el trabajo en paralelo.
  - **Ejemplos de módulos:** Elementos arquitectónicos base (paredes, suelos), detalles arquitectónicos (puertas, ventanas, escaleras), mobiliario (sillas, mesas, sofás), y exteriores (terreno, vegetación).
- **Archivos .blend Separados por Módulo:**
  - Cada componente modular tendrá su propio archivo .blend (ej., mobiliario_cocina_mesa.blend). Esto evita que varios trabajen directamente en un mismo archivo maestro grande, previniendo conflictos.
- **Funciones Clave en Blender para Trabajo en Equipo:**
  - **Append/Link:** Si necesitas usar un modelo ya terminado por otro (ej., una puerta estándar), puedes usar File > Append para importar una **copia** del objeto a tu escena. Si el modelo fuente podría cambiar y quieres que los cambios se reflejen, usa File > Link para referenciarlo. Para modelos que no se modificarán más en su archivo original, Append es más seguro.
  - **Organización en Outliner:** Mantén el Outliner de Blender limpio, con nombres claros y jerarquías lógicas (agrupa objetos relacionados bajo un

Empty). Esto es vital para que otros modeladores y el equipo del motor entiendan la estructura del modelo.
    - **Colecciones:** Usa Colecciones para agrupar objetos relacionados (ej., "Mobiliario_Cocina", "Luces_Salon"). Mejora la organización y permite ocultar/mostrar grupos fácilmente.
- **Flujo de Trabajo con Git/GitHub para Modeladores:**
    1. **Asignación de Tareas:** El Líder/Coordinador asignará tareas específicas (ej., "Modelar y texturizar el set de sofás para la sala").
    2. **Clonar/Pull:** Antes de empezar a trabajar, asegúrate siempre de tener la última versión del repositorio haciendo un git pull para obtener los cambios de los demás.
    3. **Modelar y Texturizar (Local):** Trabaja en tu archivo .blend local para la tarea asignada.
    4. **Optimización: Es crítico** que el modelo esté optimizado para rendimiento VR: bajo conteo de polígonos, buenas UVs y materiales PBR configurados.
    5. **Exportación de Activos:** Una vez terminado y optimizado, exporta el modelo a un formato compatible con Unreal Engine, preferiblemente **FBX o GLB/GLTF**. Asegúrate de que el escalado y las transformaciones sean correctos en la exportación.
    6. **Subida al Repositorio (Commit & Push):**
        - Guarda el archivo .blend (fuente) en la carpeta designada del repositorio (ej., /Blender_Sources/).
        - Guarda el archivo exportado (.fbx o .glb) en la carpeta del repositorio que el motor usará (ej., /Assets/Models/).
        - Realiza un git commit con un mensaje descriptivo de los cambios y luego un git push para subir los archivos al repositorio central en GitHub.

6. Cómo se Va a Trabajar en el Equipo de los Desarrolladores del Motor Gráfico

La colaboración en Unreal Engine es el corazón de la experiencia VR. La organización y el uso de Git son claves aquí:

- **Organización de Carpetas en el Motor:**
    - Es crucial establecer una estructura de carpetas lógica y consistente dentro del proyecto de Unreal Engine.
    - **Estructura Base:** Assets/Models, Assets/Materials, Assets/Scripts, Assets/Scenes, Assets/Prefabs, Assets/Textures, Assets/Audio, Assets/UI.
    - **Subcarpetas:** Dentro de cada categoría, crea subcarpetas para mayor orden (ej., Assets/Models/Architecture, Assets/Models/Furniture).
- **Convenciones de Nombres de Objetos:**
    - Sigue estrictas convenciones de nombres para mantener la interfaz del editor

limpia y facilitar la identificación de elementos.
    - **Ejemplos:** Objetos 3D: OBJ_NombreObjeto (ej., OBJ_MesaCocina); Materiales: MAT_NombreMaterial (ej., MAT_MaderaClara); Blueprints/Scripts: BP_NombreBlueprint (ej., BP_PlayerMovement); Prefabs: PFB_NombrePrefab (ej., PFB_PuertaSalon).
- **Uso Estratégico de Prefabs:**
    - Los Prefabs son la base de la reutilización y la colaboración en Unreal Engine. Permiten crear "plantillas" de objetos o grupos de objetos con su lógica y componentes ya configurados.
    - Si configuras una puerta con su lógica de apertura, la conviertes en un Prefab. Los demás pueden instanciar ese Prefab en cualquier lugar de la escena, y los cambios al Prefab original se reflejarán automáticamente en todas sus instancias. Esto permite que los desarrolladores trabajen en sistemas diferentes y luego integren sus módulos de manera controlada.
- **Uso de Control de Versiones con Git (Detallado para Unreal):**
    - **Repositorio Único:** Todo el proyecto del motor residirá en el mismo repositorio Git/GitHub que los archivos de Blender. Esto centraliza la gestión de versiones de todo el proyecto.
    - **Git Large File Storage (Git LFS): Es absolutamente esencial** para manejar los archivos grandes del motor (modelos FBX/GLB importados, texturas, archivos de audio). Git LFS no guarda el archivo grande directamente en el historial de Git, sino una pequeña referencia a él, manteniendo el repositorio de Git ligero y rápido. **Deben configurarlo para todos los tipos de archivos grandes desde el inicio.**
    - **.gitignore Esencial:** Cada motor gráfico genera archivos temporales o de caché que NO deben ser versionados (ej., la carpeta Binaries o Saved en Unreal Engine). Usen un archivo .gitignore específico para Unreal Engine para evitar que estos archivos se suban y creen problemas. Pueden encontrar plantillas confiables en línea (ej., gitignore.io).
    - **Flujo de Trabajo de Ramas:**
        - main (o develop): Es la rama principal que siempre debe estar en un estado funcional. **Los desarrolladores NO trabajan directamente aquí.**
        - **Ramas de Funcionalidad (feature/nombre-de-la-funcionalidad):** Para cada tarea grande o sistema nuevo (ej., feature/player-movement, feature/door-interactions), un desarrollador crea una rama específica y trabaja en ella de forma aislada.
        - **Commits Frecuentes:** Realicen git commit pequeños y frecuentes con mensajes claros que describan los cambios. Esto facilita el seguimiento y la reversión si es necesario.

        - **Pull Requests (PRs):** Cuando una funcionalidad en una rama está lista, se crea un Pull Request (en GitHub) para que otro miembro del equipo (idealmente el líder o un compañero) la revise. Una vez aprobada y probada, se fusiona (merge) en la rama principal.
- **Asignación de Tareas dentro del Grupo del Motor:**
    - **Integrador Principal / Diseñador de Niveles:** Responsable de importar modelos 3D, ensamblar escenas, posicionar objetos y mantener la coherencia espacial de la casa.
    - **Programador Principal de Interacciones:** Se enfoca en desarrollar la lógica de todas las interacciones del usuario (movimiento del jugador, interacción con puertas, luces, objetos recogibles, UI).
    - **Especialista en Iluminación y Optimización:** Encargado de configurar la iluminación global y local, efectos de post-procesado y optimizaciones de rendimiento a nivel de motor (occlusion culling, LODs, atlas de texturas, ajustes de calidad VR).
    - **Tester / QA de Motor:** Colabora con Documentación y QA para probar la experiencia VR, identificar bugs y problemas de rendimiento/usabilidad.

7. Cómo se Va a Trabajar Entre Ambos Equipos (Modeladores y Desarrolladores de Motor)

La colaboración fluida entre modeladores 3D y desarrolladores de motor es la clave del éxito. Se logrará a través de un ciclo continuo de entrega de activos e integración:

1. **Definición de Requisitos y Estándares (¡Desde el Inicio!):**
    - **Al inicio del proyecto**, ambos equipos se reunirán para definir estándares cruciales: unidades de medida (ej., metros), escala de los modelos, convenciones de nombres, límites de polígonos para diferentes tipos de activos, y el formato de exportación preferido para el motor (FBX, GLB/GLTF).
    - Compartiremos referencias visuales (planos, bocetos) de la casa para asegurar que todos tienen la misma visión.
2. **Ciclo de Entrega y Consumo de Activos:**
    - **Modeladores Producen y Optimizan:** Los modeladores crean, texturizan y optimizan rigurosamente los modelos 3D para VR (reducción de polígonos, UVs limpias, materiales PBR).
    - **Exportación Estándar:** Cada modelador exporta sus modelos optimizados en el formato acordado (FBX/GLB) y los coloca en la carpeta designada dentro del repositorio Git (/Assets/Models/).
    - **Push a GitHub:** Los modeladores hacen git commit y git push regularmente, subiendo tanto los archivos .blend (fuente) como los exportados.
    - **Desarrolladores de Motor "Pull":** Los desarrolladores del motor hacen git

pull de forma regular (al inicio de cada sesión de trabajo, por ejemplo) para obtener los últimos modelos y cambios de código del repositorio.
    - **Integración en el Motor:** El equipo del motor importa los nuevos modelos a las escenas correspondientes, los posiciona, aplica materiales y configura la iluminación.
    - **Implementación de Lógica:** Paralelamente, los programadores desarrollan la lógica de interacción que utilizará esos modelos (ej., un script de puerta para el modelo de puerta).
3. **Comunicación Constante y Feedback (¡CLAVE!):**
    - **Reuniones Sincronizadas:** Estableceremos reuniones semanales conjuntas para revisar el progreso, discutir desafíos y planificar las siguientes tareas para ambos equipos.
    - **Canales de Comunicación Rápidos: Discord y su Importancia:**
        - Usaremos **Discord** como nuestro medio de comunicación principal para preguntas rápidas, compartir capturas de pantalla de avances o señalar problemas.
        - **Es fundamental conectarse lo más seguido posible, idealmente cada día.** La comunicación activa y casi constante nos permitirá resolver dudas al instante, compartir progresos y evitar que los problemas se acumulen. La inactividad o la falta de respuesta rápida puede ralentizar a todo el equipo, ya que los modelos y la lógica están interconectados.
    - **Bucle de Feedback:**
        - Los desarrolladores del motor informarán a los modeladores si un modelo tiene problemas de rendimiento, si las UVs no funcionan bien con los materiales del motor, o si hay errores de escala.
        - Los modeladores pueden preguntar sobre las necesidades específicas del motor (ej., si necesitan modelos con un LOD específico).
        - El rol de Documentación y QA será crucial para canalizar y formalizar este feedback, asegurando que se corrija y se aprenda de la experiencia.
4. **Pruebas Conjuntas:**
    - Regularmente, ambos equipos deben participar en pruebas del proyecto dentro del motor, preferiblemente en un entorno VR.
    - Esto permite a los modeladores ver cómo sus activos se comportan en el juego y a los desarrolladores recibir feedback directo sobre la experiencia inmersiva.
    - Los problemas de rendimiento o visuales se identificarán más rápidamente cuando ambos equipos estén presentes.
5. **Resolución de Conflictos y Dependencias:**
    - Cuando surjan dependencias (ej., el programador necesita un modelo

específico para una interacción), la comunicación activa y la planificación del Líder/Coordinador asegurarán que el modelador priorice esa tarea.
    - Git ayudará a gestionar conflictos en el código o en los archivos del proyecto del motor. Para conflictos en modelos binarios, la comunicación sobre quién está trabajando en qué componente específico es la mejor prevención.

Al implementar estas estrategias, ambos equipos funcionarán como engranajes de una misma máquina, con una visión compartida, herramientas de colaboración robustas y un ciclo de feedback constante para lograr un proyecto de VR cohesivo y de alta calidad dentro del plazo establecido.