ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Optimización de rendimiento del Videojuego Mi Bosque 3D

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Ciencias Computacionales

Presentado por:

Matheus Javier López Vélez

Luis Andrés Villamar Saltos

GUAYAQUIL - ECUADOR

Año: 2021 - 2022

DEDICATORIA

El presente proyecto y mi carrera universitaria están dedicados enteramente a Dori Noemi Vélez García, una mujer de origen humilde que desconoce la definición de los términos cansancio y holgazanería, cuyo mayor sueño es ver a sus hijos triunfar en la vida. Una madre que en momentos de bienestar o de adversidad ha sabido fungir como ancla y como flotador según la situación lo amerite.

Matheus J. López Vélez

DEDICATORIA

Este proyecto se lo dedico a la comunidad politécnica y a la ESPOL como entidad educativa, que me permitió expandir mis horizontes, aumentar mis conocimientos además consolidar mis sueños y metas. Que este proyecto sirva para mejorar la experiencia de usuario e incremente el alcance de los videojuegos como herramienta educativa.

Luis A. Villamar Saltos

AGRADECIMIENTOS

Agradezco a todas aquellas personas que llegaron a mi vida para generar cambios positivos. A aquellos amigos de antaño cuya lealtad se ha mantenido férrea a través de los años y aquellos amigos de hogaño que llegaron a ofrecer su apoyo incondicional y desinteresado. A los colegas con los cuales se comparte una meta en común y un mismo camino a seguir. A aquellos profesores que fueron más allá de lo que su rol dictaba para ofrecer una formación integral. Y finalmente a la familia, que sin importar el contexto geográfico, social o económico siempre están para brindar una mano. Todos tienen mis más sinceros agradecimientos.

Matheus J. López Vélez

AGRADECIMIENTOS

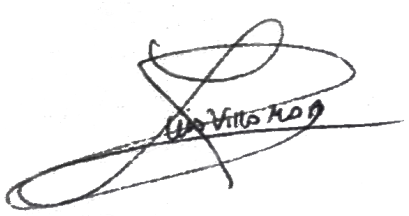
Agradezco de todo corazón a mi padre, madre y hermana por haberme apoyado incondicionalmente en el transcurso de la carrera, y antes de ella. También estoy agradecido con todos mis compañeros y amigos que he obtenido a lo largo de la carrera, ya no solo por hacer más amena la experiencia de aprender y educarse sino también por su apoyo a nivel personal. Además de un agradecimiento especial a mis profesores que ejercieron de una manera magistral su clase y siempre me motivaron a aprender más.

Luis A. Villamar Saltos

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Matheus Javier López Vélez y Luis Andrés Villamar Saltos damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Matheus Javier López Vélez



Luis Andrés Villamar Saltos

EVALUADORES

LUCIA MARISOL VILLACRES FALCONI
Digitally signed by LUCIA MARISOL VILLACRES FALCONI
Date: 2022.04.04 12:58:24 -05'00'

PhD. Lucía Marisol Villacrés Falconi

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
RODRIGO ALEXANDER SARAGURO BRAVO

MSc. Rodrigo Alexander Saraguro Bravo

PROFESOR TUTOR

RESUMEN

Ecuador es un país con gran biodiversidad de especies y, ecosistemas y el tema del cuidado ambiental está en continuo auge en los recientes años dando así a una cultura ambiental prometedora dentro de nuestra sociedad. Por esto la ESPOL ha desarrollado el videojuego Mi Bosque 3D sin embargo el estado actual del videojuego consume una gran cantidad de recursos computacionales y la experiencia percibida en la jugabilidad se ve afectada por problemas de rendimiento que causan falencias y ralentizaciones. Se analizó el rendimiento del videojuego Mi Bosque 3D registrando los FPS en el transcurso de su ejecución. A partir de lo cual se detectaron los puntos de incidencia con menor rendimiento y sus orígenes. Para reducir la cantidad de elementos en ejecución de segundo plano, se implementó un administrador de componentes encargado de instanciar o destruir elementos necesarios según la proximidad del jugador. Para reducir la carga gráfica se implementó un administrador gráfico que le permita al jugador personalizar los componentes estéticos del videojuego según sus preferencias. Finalmente se mejoró la integración entre el sistema de logros con la clase de estadísticas y peticiones web. A partir de las implementaciones se detectó una mejora promedio del 35% y una mejora máxima del 55% en el rendimiento del videojuego. Los nuevos módulos logran estabilizar y normalizar los FPS de manera exitosa a través del manejo de recursos.

Palabras Clave: Optimización de videojuegos 3D, Unity, FPS, Rendimiento, Mesh, Prefab, Draw calls.

ABSTRACT

Ecuador is a country with great biodiversity of species, ecosystems and the subject of environmental care has been on the rise in recent years, thus giving a promising environmental culture within our society. For this reason, ESPOL has developed the video game Mi Bosque 3D; however, the current state of the game consumes a large amount of computational resources and the perceived gameplay experience is affected by performance problems that cause shortcomings and slowdowns. The performance of the video game Mi Bosque 3D was analyzed by recording the FPS during its execution. From which the points of incidence with lower performance and their origins are detected. To reduce the number of items running in the background, a component manager has been implemented to instantiate or destroy needed items based on the player's proximity. To reduce the graphic load, a graphic manager was implemented that allowed the player to customize the aesthetic components of the game according to their preferences. Finally, the integration between the achievement system with the statistics class and web requests has been improved. From the implementations, an average improvement of 35% and a maximum improvement of 55% in the performance of the game was detected. The new modules successfully stabilize and normalize FPS through resource management.

Keywords: 3D Video Game Optimization, Unity, FPS, Performance, Mesh, Prefab, Draw calls.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL: Escuela Superior Politécnica del Litoral

VR: Virtual Reality

AAA: Calidad Triple A

UI: User Interface

MIT: Massachusetts Institute of Technology

JIT: Just in time

AOT: Ahead of time

Boehm GC: Boehm garbage collector

TSL: Transport Layer Security

FPS: Frames per second

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1.1 Introducción

En las últimas décadas se ha vuelto un hecho incuestionable el arraigo que tiene la tecnología en el diario vivir al punto en el cual no se puede discernir con certeza si esta herramienta es una conveniencia para facilitar tareas o una necesidad contemporánea (Paullet et al., 2010). La revolución tecnológica es percibible a través de los diversos sectores económicos y sociales. Claros ejemplos son la abundancia de páginas web y anuncios destinados a publicitar productos y servicios, plataformas de ventas como Amazon que incentivan la comercialización de productos, servicios de streaming que ofrecen acceso a una inmensurable cantidad de entretenimiento a la familia desde el hogar o inclusive plataformas que contienen cursos en línea para facilitar la educación desde cualquier lugar sin restricciones de horarios.

Al enfocar el uso de la tecnología en el sector educativo, las últimas tendencias han apostado a la ludificación, también conocida como “Gamificación”. Este principio sugiere usar mecanismos, dinámicas y marcos de los juegos para incentivar el aprendizaje (Li, Dong, Untch y Chasteen, 2013). En otras palabras, usar videojuegos de naturaleza entretenida para proporcionar material educativo al jugador sin que este pierda el interés. Bajo este principio la Escuela Superior Politécnica del Litoral inició el proyecto Mi Bosque 3D que consiste en el desarrollo de un videojuego educativo en el cual el jugador será capaz de explorar el Bosque Protector Prosperina interactuando con la flora y fauna endémica. La importancia del proyecto es de mayor relevancia en la actualidad debido a la presente emergencia sanitaria originada por el virus COVID-19 y sus variantes ya que el videojuego permitirá a niños y jóvenes adquirir conciencia ambiental sin exponerse a interacciones innecesarias.

1.2 Descripción del problema

Mi Bosque 3D es un videojuego de libre uso que actualmente está enfocado a ser distribuido en instituciones públicas, con sus principales beneficiarios siendo niños, jóvenes y profesores. Puede ser ejecutado en dispositivos portátiles como celulares,

tablets, laptops y en equipos fijos como computadoras de escritorio. En términos de funcionalidad, la versión actual del producto ha sido probada con éxito y con gran acogida por parte de niños en unidades educativas en las ciudades de Guayaquil, Durán y El Triunfo. Sin embargo, en los equipos de menor gama tecnológica se presentaron dificultades de jugabilidad que causaron a su vez un deterioro en la experiencia percibida por el jugador.

En la versión actual de Mi Bosque 3D se requiere recursos informáticos mínimos por tal motivo ha sido testeado en dispositivos de gama alta o media. Según testimonios recolectados, varios dispositivos de gama baja mostraron ralentizaciones visuales durante pruebas del videojuego, momentos en los que se traba totalmente toda funcionalidad, sobrecalentamiento del dispositivo al recorrer el escenario 3D y una respuesta tardía al presionar ciertos botones de la interfaz gráfica. Estos inconvenientes degradan la calidad general de la experiencia interactiva y en los casos más extremos generan desagrado y falta de interés. Las retroalimentaciones de falencias percibidas tienden a centrarse en instituciones públicas de bajos recursos y en usuarios con hogares de bajos ingresos.Es por ello que mejorar la optimización y rendimiento del videojuego posibilitará su uso y goce en una mayor variedad de dispositivos.

1.3 Justificación del problema

Considerando la problemática mencionada desde una postura socio-económica, el videojuego está dirigido a instituciones públicas; sin embargo, los dispositivos capaces de ejecutar el software sin inconvenientes son aquellos de gama alta. Según datos recientes recopilados por el Instituto Nacional de Estadísticas y Censos, la población del Ecuador está conformada en un 25% por la clase social media, 38% por la clase social baja y 20% por sujetos en pobreza extrema (Ecuadorencifras, 2021). El estado económico actual del país y de su población se vió deteriorado sobre todo en los últimos años por el impacto global que tuvo el virus COVID-19. Dicho impacto incluye situaciones como pérdidas de empleo, el incremento de gastos en insumos

médicos y de higiene personal, la necesidad de vender artículos no esenciales y en general una reducción en la calidad del estilo de vida para muchas familias. Es por lo cual se puede asumir que la mayor parte de los usuarios no siempre dispondrán de dispositivos de último modelo.

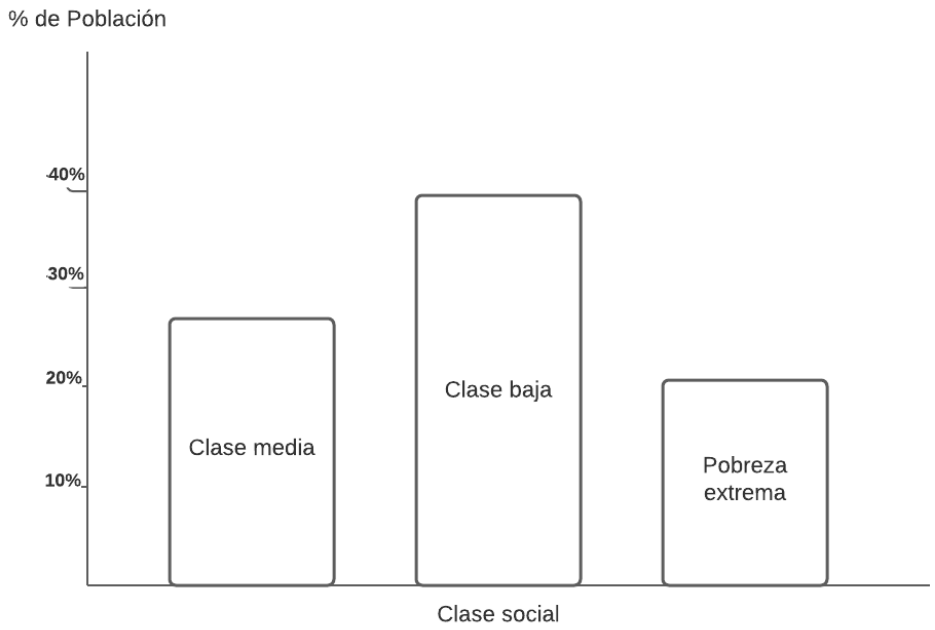


Figura 1.3.1: Distribución de clases sociales media y baja en Ecuador

Por este motivo la optimización del videojuego 3D permitirá a más dispositivos ejecutar el videojuego, facilitando así su uso en computadoras o dispositivos con menor capacidad de hardware, además de equiparar las posibilidades de acceso al videojuego y mejorar la experiencia para todos los usuarios.

1.4 Objetivos

1.4.1 Objetivo general

- Optimizar el rendimiento del videojuego educativo Mi Bosque 3D mediante la implementación de módulos de gestión de recursos para mejorar la experiencia didáctica del usuario y el aprendizaje

1.4.2 Objetivos específicos

- Registrar los puntos de incidencia en los que el videojuego refleja bajo rendimiento.
- Implementar un manejador de objetos para administrar los elementos de las estaciones según la proximidad y rango de visión del personaje.
- Documentar estándares de programación y arquitectura limpia para las implementaciones futuras.

1.5 Marco teórico

En esta sección se detalla el contexto en el que está desarrollado el videojuego Mi Bosque 3D, incluyendo sus características y el motor gráfico en el que está desarrollado. También se describen los factores principales que afectan la calidad y el rendimiento de los componentes en las ejecuciones a tiempo real del videojuego y las herramientas útiles para la cuantificación de las métricas relacionadas. Finalmente se incluye una revisión de trabajos similares en el campo del consumo y optimización de recursos en el motor gráfico Unity.

1.5.1 Mi Bosque 3D

El videojuego educativo Mi Bosque 3D es un videojuego del género simulación que sitúa al jugador en un ambiente lleno de flora y fauna con la cual puede interactuar de forma segura. El jugador podrá caminar, correr, saltar mientras explora y aprende en el proceso. El objetivo del videojuego es recorrer el sendero al mirador en la cima del Bosque Protector Prosperina mientras cumple misiones de guardabosque ayudando animales, sembrando árboles e inclusive generando conciencia ambiental con temas de reciclaje y medidas de seguridad. Como sistema de incentivo al jugador, su avance se ve proyectado en la obtención de medallas de guardabosque, progreso en la lista de misiones por cumplir y al finalizar con un certificado digital personalizado de guardabosque honorario.

Las locaciones centrales del videojuego son el tutorial, el lobby y el bosque. En el tutorial el jugador es capaz de aprender los comandos básicos que utilizará durante el videojuego. En el lobby el jugador tendrá acceso a material educativo en forma cabañas que proyectan en su interior videos o poseen exposiciones sobre las distintas especies. Finalmente, en el bosque tendrán el recorrido interactivo del sendero al mirador donde recibirán misiones a realizar y preguntas que validen los conocimientos adquiridos.

En su versión actual el videojuego cuenta con 7 desafíos, 8 misiones, aproximadamente 50 preguntas relacionadas al bosque, 13 especies de animales y 6 especies de árboles con sus respectivos modelos, imágenes y datos biológicos. Los ejecutables realizables son compatibles en dispositivos con sistema operativo Windows, macOS. iOS y Android con jugabilidad en teclado/mouse o pantallas táctiles.

1.5.2 Unity

Unity es un motor gráfico bastante conocido principalmente debido a su fácil acceso, variedad de cursos y su comunidad que colabora activamente en temas relacionados con el desarrollo de videojuegos. En el 2021 ha alcanzado el 61% de preferencia como motor gráfico a los desarrolladores encuestados y posee el segundo lugar con respecto a los ámbitos móviles y web (Unity Gaming Report, 2021).

1.5.2.1 Arquitectura

El motor Unity está formado de manera nativa con los lenguajes de programación C y C + + de manera interna. A usuarios ofrece una capa de abstracción superior en el lenguaje C# para facilitar la programación. En su estructura unity integra el framework .NET para garantizar que los productos resultantes en el motor puedan ser ejecutados en una gran variedad de dispositivos y como backend para los scripts usa los framework “Mono” para compilar en tiempo de ejecución según la demanda (JIT) y IL2CPP para compilar la aplicación entera previa a la ejecución (AOT). El editor de Unity como tal usa ambos backends y gracias a esto es posible ejecutar pruebas y

editar variables a tiempo real. Al momento de realizar el ejecutable se puede elegir cuál de los dos backends persistirá, siendo Mono el de más rápida compilación y con independencia de plataforma.

Para evitar peso innecesario en archivos binarios, el framework IL2CPP remueve líneas de código no utilizadas al compilar el ejecutable de forma automática, mientras que en Mono debe seleccionarse la opción. Como recolector de basura Unity utiliza el Boehm GC para limpiar la memoria ocupada debido a que este colector está escrito en C y C++ al igual que Unity. Sin embargo, dado que los scripts realizados por los desarrolladores son escritos en C#, abran ciertos tipos de objetos no recogibles que deberían ser evitados como Type.GetMethods o Assembly.GetTypes y la clase WeakReference. El recolector puede ser desactivado y configurado pero no es recomendado. La manera en la que funciona el recolector de basura es que al destruir un objeto instanciado de forma nativa en C++ y romper la relación con su contraparte instanciada en C#, el recogedor se encarga de eliminar dicha contraparte.

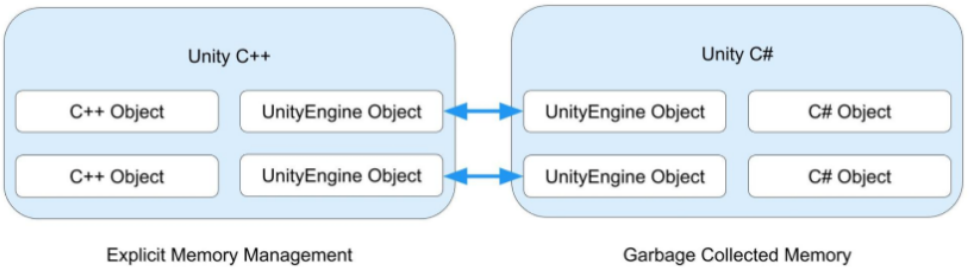


Figura 1.5.3.1: Relación de Unity Game Object en C++ y C# (Unity Documentation, 2020)

Unity es capaz de implementar librerías externas de .NET tomando en consideración que deben ser exhaustivamente probadas para asegurar su funcionalidad. Para su componente en línea, Unity dispone de soporte TSL para garantizar la seguridad en la capa de transporte. Para realizar tareas paralelas, Unity implementa su “Sistema de trabajos”, en lugar de los tradicionales hilos, para evitar condiciones de carrera. A pesar de ser capaz de utilizar hilos, la documentación de Unity se opone a dicha práctica e incentiva seguir la opción más segura.

1.5.2.2 Clasificación y eficiencia de los componentes

Unity cuenta con una amplia variedad de componentes disponibles para la creación de videojuegos abarcando desde videojuegos 2D, pasando por 3D y culminando en VR. Incluso con renderización basada en tiempo real el cual aumenta la fidelidad de la inmersión del videojuego. Para lo cual la forma de utilizar estos componentes es de vital importancia para determinar los requisitos mínimos para que el videojuego pueda ser ejecutado en la mayor cantidad de dispositivos posibles sin comprometer la calidad gráfica del mismo. Entre los componentes que ofrece Unity para las diversas necesidades de un videojuego existen varias alternativas para lograr el mismo funcionamiento pero con ciertas diferencias tanto en el apartado de rendimiento como en realismo del entorno 3D que se plantea representar en el videojuego. Los componentes de mayor impacto en el rendimiento de ejecución de Unity son:

- Scenes:

  Las escenas son los contenedores de todos los objetos de una parte o todo el videojuego los cuales involucran desde objetos, texturas, ambientes, iluminación, cámara entre otras. (Unity Documentation Scene). Siendo que el rendimiento de las escenas dependerá de dos tipos de factores: Factores propios y Factores de rendimiento.

  Para reducir los factores propios en incidencias de rendimiento se puede considerar el tamaño de la escena y sus métodos de carga. Las escenas pueden ser cargadas de forma síncrona paralizando el videojuego hasta que se cargue la escena o asíncrona realizando la carga en segundo plano y también pueden ser cargadas de forma simultánea permitiendo la coexistencia de múltiples escenas a la vez o de forma secuencial cargando la siguiente y destruyendo la anterior.

Figura 1.5.4.1: Escena de bosque en Mi Bosque 3D

- Game Object: Cualquier objeto en Unity es categorizado como Game Object. Estos poseen un componente intrínseco el cual es Transform, este componente determina la posición, rotación y escala del gameobject. (Unity Documentation GameObject). Mientras más cantidad de gameObject existan, mayor serán los objetos que Unity tendrá de dibujar en el entorno en caso de ser visibles.

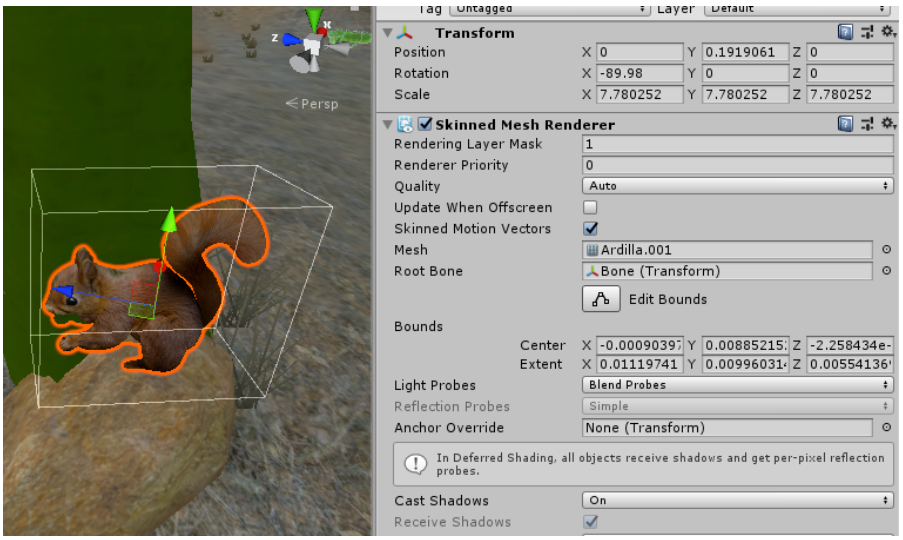


Figura 1.5.4.2: Ardilla en Mi Bosque 3D

- Lightning: Unity provee varias maneras de representar la luz en los escenarios, entre ellas están las luces en tiempo real, globales y los mapas de luces o predefinidos las cuales son más eficientes y menos realista que las luces en tiempo real. (Unity Documentation Light). Debido a que la iluminación se calcula mediante vectores representando luces, reflexiones y refracciones mientras realiza cambios visuales en los valores de color de la textura impactada y creando sombras de ser necesario. Cuando la iluminación se define como estática, estos cálculos se realizan una vez para dicha iluminación, pero para luces dinámicas y objetos generadores de sombras que se muevan los vectores se analizan y calculan constantemente.
- Models: Es un archivo que contiene forma y apariencia de los objetos 3D en Unity, involucrando mesh, texturas y materiales. (Unity Documentation Models). Debido a que son una representación 3D de objetos en la realidad digital. La manera en la que un software grafica un mesh es a partir de la unión de superficies, ya que son un conjunto de arreglos de vértices y triángulos que determinan la composición de un objeto (Unity Documentation Meshes). A mayor número de superficies, más precisa será la representación visual de un objeto 3D, pero requerirá mayor esfuerzo computacional para calcularlas. Es por ello que se tiende a sacrificar calidad para mejorar el rendimiento. En el caso que un modelo esté compuesto por múltiples componentes, se opta por unificar dichos componentes para formar uno solo que requiera menos callback por frames. Utilizar un solo material con texturas unificadas también reduce los callbacks. Y por último agrupar modelos en batches también reduce los callbacks requeridos.

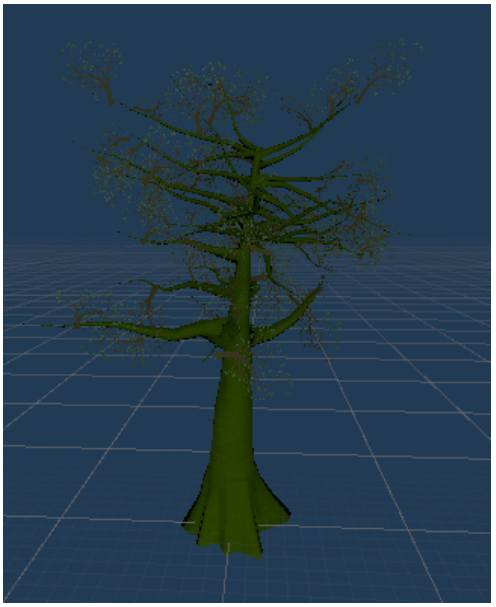


Figura 1.5.4.3: Modelo de árbol en Mi Bosque 3D

- Scripts: Son archivos necesarios para responder a los eventos del videojuego y la entrada de acciones del jugador, forman la parte crucial en la experiencia del videojuego permitiendo crear o modificar los efectos gráficos y las físicas de este. (Unity Documentation Scripting). Además de que su influencia tiene impacto en el rendimiento general; sin embargo si son implementados siguiendo las buenas prácticas junto con la experticia de los programadores participando en el desarrollo se puede reducir tal impacto en el videojuego. Conocimientos de refactorización, normalización, buenas prácticas de programación, malos olores y los pilares de la programación pueden encaminar al correcto desarrollo de la lógica del videojuego.

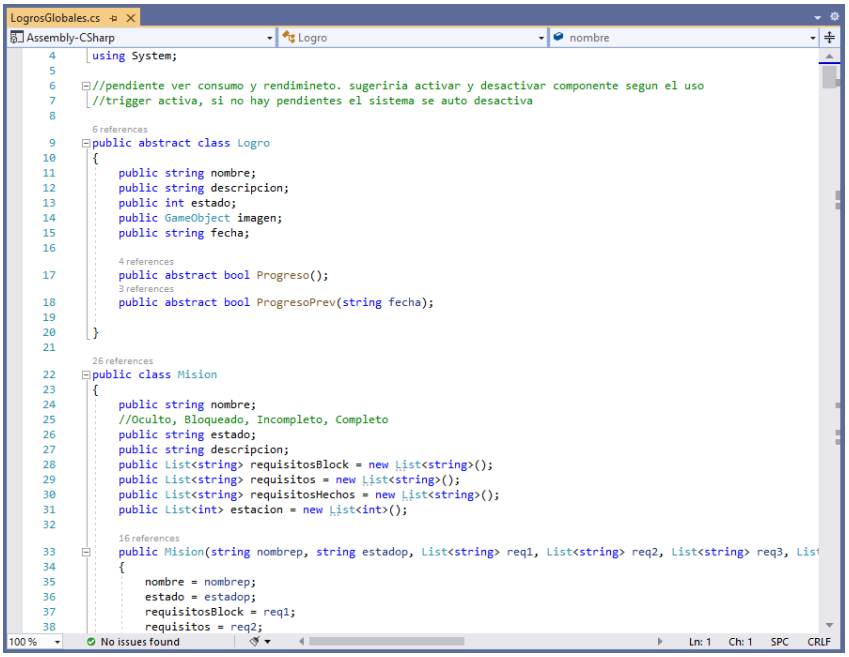


Figura 1.5.4.4: Script C# en Mi Bosque 3D

- Camera: Es la perspectiva del jugador, normalmente en entornos 3D la cámara aplana la vista usando el efecto de perspectiva similar como lo hacen los ojos humanos (Unity Documentation Camera). Poseer un objeto cámara en el videojuego es fundamental para la experiencia de videojuego, sin embargo tener múltiples objetos cámara supondría un impacto al rendimiento del videojuego.

1.5.2.3 Manejo de estados

Debido a la naturaleza de los videojuegos, lo que se diseña es una simulación en la cual cada objeto maneja estados similares a los que poseería el mismo objeto en la realidad. Los estados más comunes son: poseer una posición relativa en el espacio, una orientación, ser afectados por fuerzas externas o poseer resistencia y dureza propia definida, entre otros. Y a lo largo del tiempo analiza si hay factores externos intentando alterarlo para reflejar algún cambio. La arquitectura de la clase “Monobehaviour” está diseñada para imitar todas estas cualidades de objetos reales y si bien esto suena ideal, todos estos métodos y atributos estarán disponibles en el caso

de ser necesarios. Considerando que cada objeto instanciado contará con sus propios estados, atributos y métodos, los objetos monobehaviour pueden llegar a ocupar una gran cantidad de recursos informáticos.

La primera solución ante la detección de mal rendimiento es el uso de “Prefabs” no instanciados. Un prefab representa una plantilla de un objeto de naturaleza monobehaviour y puede ser fácilmente instanciado cuando se lo necesite, evitando tenerlo ocupando memoria durante la ejecución entera. Sin embargo, para objetos simples sigue representando un consumo innecesario de recursos. La segunda solución se considera usar clases estáticas, pero al no ser serializables, el inspector de Unity no permite la visualización y edición de sus variables requiriendo de ajustes personalizados para realizar dichas acciones. Además, las clases estáticas son volátiles y pierden su estado cada vez que la ejecución se reinicia. Según Unity, la solución más óptima para manejar estados y otros atributos de objetos básicos es el uso de la clase “Scriptable Object”. Los scriptable objects almacenan información compartida útil para objetos Monobehaviour y pueden funcionar sin la necesidad de ser instanciados como objetos comunes, cuentan solamente con métodos suficientes para subsistir en el contexto de Unity y los datos/cambios de sus estados se mantienen de ejecución en ejecución.

1.5.3 CPU

El CPU es un recurso valioso debido a que gracias a este componente de hardware se puede realizar cálculos a una gran velocidad, debido a esto el rendimiento de un videojuego está muy ligado a la cantidad de trabajo que el CPU pueda realizar para mantener la fluidez del videojuego. En el caso de Unity no es la excepción, existen diferentes cargas al CPU por cada componente de Unity, mientras más componentes renderice Unity, más cálculos y procesos deberá realizar el CPU para mantener un rango aceptable los fps.

Cada componente requiere tiempo de procesamiento por lo que para reducir el consumo del CPU (Unity Documentation OptimizingGraphicsPerformance) se realizan ciertas prácticas como las definidas a continuación:

- Combinar los objetos pequeños en objetos grandes.
- Usar atlas para evitar usar texturas individuales.
- Evitar renderizaciones en tiempo real como lo son las luces o sombras.
- Reducir el framerate en tiempos donde no se demande una actualización constante de la escena.

Aunque existen ciertos casos donde combinar objetos no representa una mejora en el rendimiento como por ejemplo son los pixel lights, en los cuales agruparlos no mejora el rendimiento en general y en los scripts donde se depende de otros factores como el uso de eventos, renderizaciones, entrada de datos o ciclos eficientes donde se reduzca el tiempo de operación del CPU.

1.5.4 GPU

La GPU es el procesador encargado de manejar y acelerar la representación de los gráficos. En el contexto del desarrollo de videojuegos, el GPU se encarga de las renderizaciones multimedias como la graficación de sprites 2D y modelos 3D, la renderización de texturas y luces, la transparencia de los objetos entre otras tareas. Dado que la representación de gráficos digitales no es más que un procesamiento de cómputo, naturalmente el CPU es el encargado original de realizar este tipo de tareas sobre todo cuando el equipo no dispone de una tarjeta gráfica dedicada. Para dispositivos que sí dispongan de tarjetas gráficas lo que se busca es balancear la carga del videojuego tanto en el CPU como el GPU.

Muchas de las prácticas de optimización dirigidas al CPU aplican al GPU, pero las prácticas más enfocadas al rendimiento gráfico son:

- Rebalancear las cargas entre CPU y GPU
- Reducir el número de materiales utilizados en los objetos (comprimir texturas)

- Reducir el número de polígonos de los objetos (resolución)
- Ajustar el nivel de detalle de los objetos según la cercanía al jugador
- Agrupar en batches de renderización objetos que comparten materiales

1.5.5 Herramienta de medición de rendimiento y datos analizados

Según estudios realizados en la “Politechnika Wroclawska” se ha analizado cómo Unity ha sido utilizado incluso para representación espacial usando datos geográficos de OpenStreetMap el cual es un proyecto colaborativo para la creación de mapas de libre uso, entre otros SDK que permiten modelar estructuras basadas en la realidad usando datos de geolocalización. Conservando rendimiento según qué SDK se utilice para el modelado, con sus beneficios y desventajas que provee cada SDK con respecto al otro, todo bajo el motor gráfico de Unity. (Sarper TEMEL, 2019)

Cuando se analiza la optimización de rendimiento en dispositivos móviles, se aplican muchas técnicas en común a las optimizaciones en dispositivos de escritorio. Jerry Blåfield en su tesis “Optimizing mobile games in a Unity environment” enfatizó el hecho de que en ocasiones, cambios individuales pueden ocasionar mejoras de rendimiento menores. Pero al juntar diversas técnicas de mejora de rendimiento los resultados pueden volverse notoriamente más significativos. (Jerry Blåfield, 2021)

Tomando en consideración trabajos previos realizados en campos similares, es percibible el potencial que tiene la combinación de aplicar diversas técnicas de optimización de forma simultánea. La correcta aplicación de técnicas y estándares en conjunto al ajuste de la calidad de resolución estética deberían aumentar de forma eficaz el rendimiento general y por ende mejorar la experiencia de usuario final.

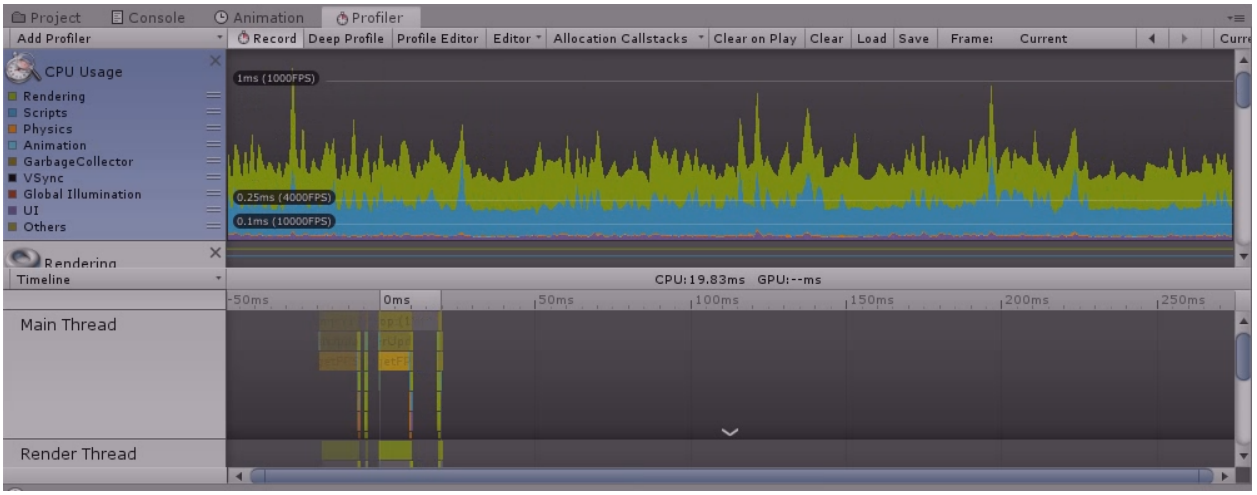


Figura 1.5.5: Profiler en Mi Bosque 3D

1.5.6 Trabajos similares

Según estudios realizados en la “Politechnika Wroclawska” se ha analizado cómo Unity ha sido utilizado incluso para representación espacial usando datos geográficos de OpenStreetMap el cual es un proyecto colaborativo para la creación de mapas de libre uso, entre otros SDK que permiten modelar estructuras basadas en la realidad usando datos de geolocalización. Conservando rendimiento según qué SDK se utilice para el modelado, con sus beneficios y desventajas que provee cada SDK con respecto al otro, todo bajo el motor gráfico de Unity. (Sarper TEMEL, 2019)

Cuando se analiza la optimización de rendimiento en dispositivos móviles, se aplican muchas técnicas en común a las optimizaciones en dispositivos de escritorio. Jerry Blåfield en su tesis “Optimizing mobile games in a Unity environment” enfatizó el hecho de que en ocasiones, cambios individuales pueden ocasionar mejoras de rendimiento menores. Pero al juntar diversas técnicas de mejora de rendimiento los resultados pueden volverse notoriamente más significativos. (Jerry Blåfield, 2021)

Tomando en consideración trabajos previos realizados en campos similares, es percibible el potencial que tiene la combinación de aplicar diversas técnicas de optimización de forma simultánea. La correcta aplicación de técnicas y estándares en

conjunto al ajuste de la calidad de resolución estética deberían aumentar de forma eficaz el rendimiento general y por ende mejorar la experiencia de usuario final.

CAPÍTULO 2

2 Metodología

2.1 Análisis primario

Para optimizar el videojuego Mi Bosque 3D es necesario conocer el contexto en el que se encuentra el videojuego. Además de que el rendimiento tiene un efecto notable para los usuarios y su experiencia de videojuego.

a) Usuarios del sistema

Los usuarios objetivos identificados del videojuego Mi Bosque 3D, son los siguientes:

- Cualquier usuario que entre en la categoría de estudiante menor de edad con una interacción mínima requerida con el ordenador
- Profesionales de la educación que tengan interés en el cuidado ambiental del bosque.

b) Percepción del usuario

Según experiencias de usuarios narradas por voluntarios pertenecientes a instituciones educativas públicas, el videojuego Mi Bosque 3D ofrece una la experiencia educativa divertida, sin embargo, se reportan ciertos puntos de lentitud notorios dentro del videojuego interrumpen la fluidez en la jugabilidad y entorpecen la inmersión del videojuego.

c) Especificaciones Técnicas

Para las pruebas de rendimiento y medición de los mismos se utilizó un ordenador con las siguientes especificaciones técnicas:

- Procesador Core I5 4 Gen
- 8Gb Ram DDR4
- Gtx 750 ti
- Disco Sólido 250 GB

- Windows 10

d) Herramienta de análisis y datos a considerar

Para determinar el rendimiento actual del videojuego Mi Bosque 3D, se realizó un análisis de métricas en tiempo real utilizando la herramienta integrada de Unity llamada Profiler. La cual nos muestra que existen momentos en los que se sufren una caída en su rendimiento, afectando así a la jugabilidad del mismo, esto es influenciado por diversos factores por lo cual no se puede determinar una causa principal del bajo rendimiento del videojuego Mi Bosque 3D, sin embargo se pueden identificar varios factores de menor impacto que impactan en la optimización en la velocidad de ejecución del videojuego.

Las métricas disponibles para análisis se encuentran: CPU, GPU, Scripts, Rendering, Physics, Global Illumination, UI, entre otros.

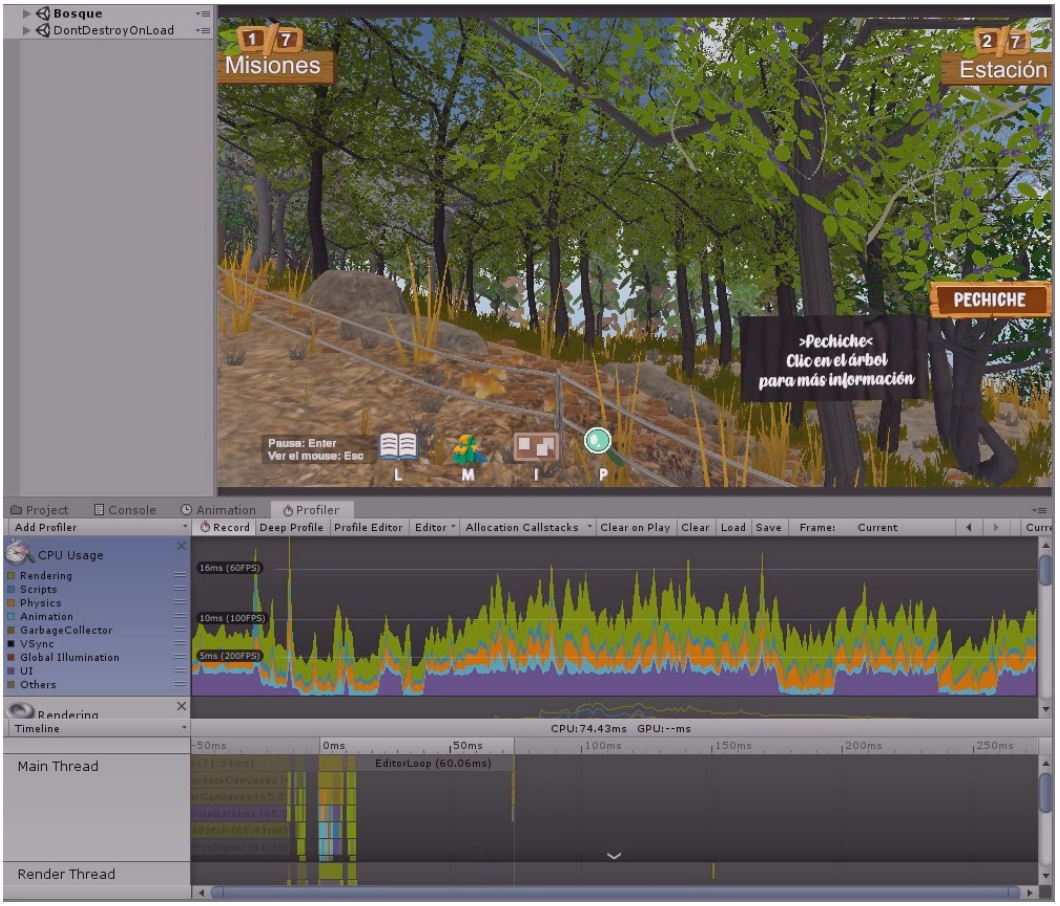


Figura 2.1.1: Métricas Observables

e) Estado actual

Actualmente el bosque está compuesto por 7 estaciones secuenciales, cada estación posee especies de flora y fauna, un punto de referencia inicial, los componentes de un desafío único por escena, elementos visualmente decorativos, paredes invisibles que restringen el paso del jugador y un generador de preguntas. Todos estos componentes existen de forma simultánea en el contexto de la ejecución del videojuego y muchos de los componentes contienen atributos y/o funcionalidades ejecutándose en segundo plano sin ser indispensables para el jugador en todo momento.

En la jerarquía de objetos del editor de Unity se pueden encontrar objetos sueltos y desorganizados que si bien no afectan a la funcionalidad, presentaron dificultades al momento de localizarlos para editar sus propiedades como localización en el espacio de la escena o su relación con otros objetos.

En la sección anexa al final del presente documento [ref] se enlistan los instantes que presentaron cambios en el rendimiento general con datos relacionados como la duración del evento (ms), el origen del evento, el tipo de incidencia y los recursos relacionados. Los puntos críticos donde las falencias son mayormente experimentadas son:

- A lo largo del recorrido completo del bosque
- En la segunda mitad del recorrido del bosque
- Al culminar las misiones 3 en adelante

2.2 Requerimientos del proyecto

Los requerimientos del proyecto describen las características y los comportamientos planteados a alcanzar y de la misma manera proponen limitaciones realistas que permitan que los objetivos generales y específicos sean realizables durante el periodo de investigación e implementación del proyecto. Los presentes requerimientos son establecidos por mutuo consentimiento entre el tutor/representante de la institución en función de cliente y los autores del presente documento,

responsables de la implementación de los cambios realizados al videojuego Mi Bosque 3D y en consideración del resultado del análisis de rendimiento antes mencionado.

2.2.1 Funcionales

Los requerimientos funcionales representan las actividades que los usuarios son capaces de realizar en el software a través de sus interacciones. En el presente contexto se definieron los siguientes requerimientos funcionales:

a) **La preservación de funcionalidades existentes:** Las nuevas implementaciones realizadas a los sistemas internos del videojuego Mi Bosque 3D no deberán interferir con las funcionalidades previamente desarrolladas. En el caso de presentar interferencias con funciones previas deberán realizarse los parches respectivos.
b) **La instanciación y destrucción de elementos necesarios en la escena:** El usuario según su ubicación en la escena y según su alcance de visibilidad será capaz de definir los componentes de las distintas estaciones estrictamente necesarios para su jugabilidad.
c) **El ajuste de la calidad visual en el videojuego:** El usuario tendrá acceso a opciones de edición de calidad gráfica que ajusten la calidad y cantidad de los elementos visibles durante la jugabilidad.

2.2.2 No funcionales

Los requerimientos no funcionales representan características del software que no necesariamente recaigan en la funcionalidad que le pueda dar el usuario, pero si en las necesidades que pueda tener el dispositivo en el que es ejecutado. En el presente contexto se definieron los siguientes requerimientos no funcionales:

a) **Mejorar la fluidez general de la jugabilidad:** Optimizar el videojuego a través de la reducción del consumo de CPU/GPU, aumentando de manera efectiva la velocidad en la que se ejecuta la jugabilidad.

b) **Reducir los tiempos de respuesta:** Agilizar la velocidad de retroalimentación en actividades como las notificaciones de logros conseguidos por el usuario al cumplir sus misiones.
c) **Estandarizar la estructura del código:** Reorganizar la estructura de los componentes del videojuego para facilitar la sostenibilidad del software y la colaboración de futuros desarrolladores.

2.2.3 Alcances

Los alcances representan limitaciones establecidas para las implementaciones a realizar. Los presentes alcances están definidos a partir de limitaciones de tiempo, recursos u oportunidades necesarias y permiten mantener una visión realista de los cambios. En el presente contexto se definieron los siguientes alcances:

a) **Las pruebas serán simuladas digitalmente:** Debido a la complejidad logística de solicitar voluntarios y diversos dispositivos para probar el videojuego, las pruebas de rendimiento se realizarán con simuladores digitales que proporcionan perfiles con los datos a analizar, como por ejemplo el profiler y el stats de Unity.
b) **Los cambios realizados aplicarán a la escena "Bosque":** Siendo la escena principal del videojuego, en la que más tiempo pasa el jugador y la que más componentes posee. Las implementaciones de mejora serán enfocadas exclusivamente a dicha escena.

2.3 Planificación de desarrollo

La metodología de trabajo a seguir se basa en el estricto seguimiento del modelo en V (Element K, 2012), el cual fomenta la planificación y uso de pruebas de la mano de cada cambio e implementación realizada. El proyecto Mi Bosque 3D ha adquirido una serie de sistemas y componentes altamente relacionados entre sí, por lo que un cambio menor puede resultar en una cadena de fallas del sistema global. Es por ello que se realizarán pruebas completas del videojuego cada vez que se realicen

cambios significativos. Las pruebas incluyen el recorrido de la jugabilidad de inicio a fin, verificando el correcto funcionamiento de la lectura de datos del jugador y cumplimiento de las misiones.

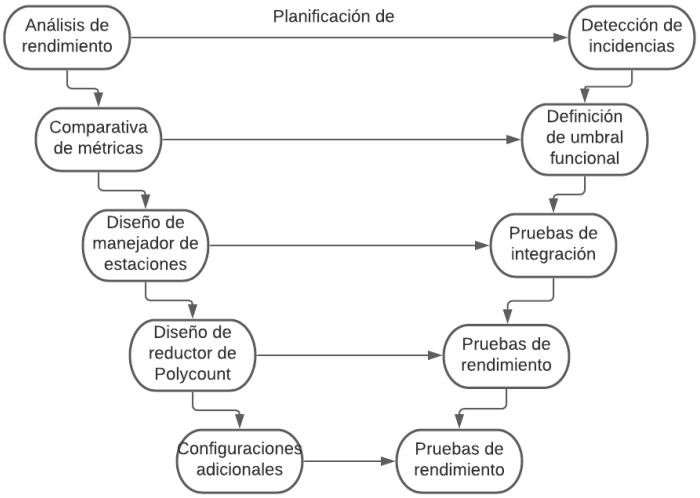


Figura 2.3.1: Flujo de trabajo en V

La implementación de los módulos también será considerada a realizarse de manera secuencial para poder analizar si generan un impacto positivo o negativo en el rendimiento general. De esta manera es posible detectar de forma temprana si alguna funcionalidad debe ser descartada o implementada. Otros motivos de descarte de funcionalidades pueden ser la falta de compatibilidad o la falta de eficiencia vs el costo de implementación.

2.4 Propuestas de solución

Entre las soluciones más comunes para la optimización del videojuego se contemplaron las siguientes opciones: la reimplementación del videojuego en un motor gráfico más adecuado para manejar recursos 3D, la separación de la versión de escritorio con la versión móvil del videojuego orientando las especificaciones al dispositivo utilizado y finalmente el modelado de nuevos modelos 3D con menor detalle que exijan menos recursos computacionales. Sin embargo, las soluciones previamente mencionadas implican desechar los años previos de desarrollo del videojuego,

duplicarle la carga laboral a futuros colaboradores o enfocarse a un área de trabajo ajena al desarrollo computacional.

Es por ello que se optó por enfocar las soluciones a implementar en 2 aspectos, los cuales son la cantidad y la calidad de recursos computacionales. Y para enfocar las implementaciones de mejora de rendimiento al sector informático evitando redundancias presentes y futuras de desarrollo se decidió desarrollar un manejador de estaciones capaz de administrar los componentes de las diversas estaciones en la escena del bosque. A través del manejador, las estaciones podrán disponer de distintos estados de mayor o menor consumo de recursos según amerite el caso.

Figura 2.4.1: Estados de las estaciones

Al iniciar una partida por primera vez, el jugador dispondrá de acceso a las funcionalidades completas de la primera estación mientras que las estaciones posteriores mantendrán un estado inicial de bajo consumo. Conforme el jugador progrese, las estaciones que vaya dejando atrás adoptaran un estado de consumo mínimo mientras las estaciones actuales activan sus funcionalidades completas. De esta manera los recursos computacionales se enfocan en las estaciones en uso. En casos de que el jugador desee retroceder a revisitar estaciones previas, el manejador habilitará únicamente los componentes necesarios y no los desafíos ya completados.

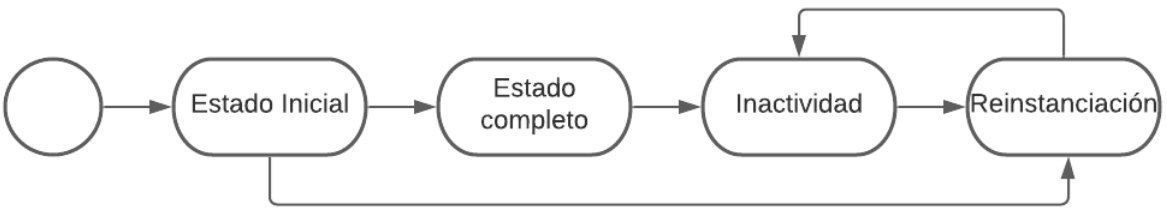


Figura 2.4.2: Flujo de estados

Finalmente el jugador tendrá acceso a una pantalla de configuraciones en la cual podrá reducir la calidad estética y gráfica del videojuego como el número de polígonos que le dan detalle a los objetos 3D o la distancia. De esta manera se reducirá el consumo general y el videojuego podrá ejecutarse con mayor fluidez en dispositivos de menor gama a la actualmente requerida.

2.5 Diseños de arquitectura

2.5.1 Modelo conceptual

La estructura de la solución conservará un esquema similar al del proyecto existente, donde un sistema/manejador independiente se encuentra suelto en la jerarquía y mantiene referencias de los componentes que controlará. Su principio funcional se basa en el manejo de las estaciones a través del llamado a funciones que permitan un mayor control sobre sus componentes, permitiendo así un mejor manejo de los recursos cargados en memoria y por lo tanto reduciendo su impacto en términos de consumo.

2.5.2 Jerarquía de objetos

La arquitectura de Unity está conformada en dos componentes de desarrollo, la lógica computacional desarrollada en código C# y la estructura visual de escenarios simulados de forma digital. En ambos componentes se aplican los principios de la Programación Orientada a Objetos y la Programación Orientada a Eventos con la intención de permitir la creación de componentes que existan y funcionen como lo hacen en el mundo real. Una correcta organización de la jerarquía de objetos es necesaria para rectificar la relación y la dependencia que tienen los objetos con los demás.

Se realizaron ajustes en la jerarquía dejando en un primer nivel todos sistemas o controladores que funcionan de forma libre o independiente y el contenedor general de

los componentes del bosque. El interior del contenedor general se compone de la siguiente manera:

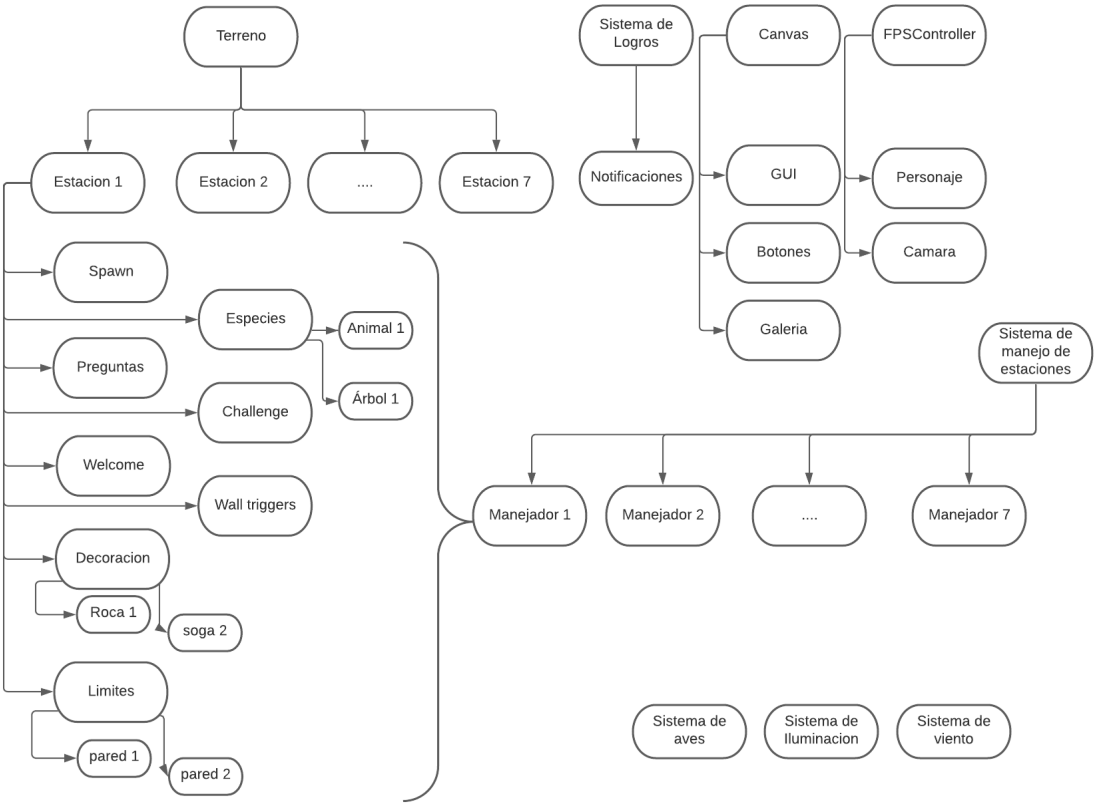


Figura 2.5.2.1: Diagrama de Objetos en escena “Bosque”

Los cambios propuestos permitirán:

- Una mejor organización para futuras implementaciones
- La creación de Prefabs unificando componentes similares relacionados
- La detección de componentes hermanos/hijos

2.5.3 Diagrama de clases del manejador de estaciones

El manejador general de las estaciones controlará la ubicación actual del jugador con la cual será capaz de discernir según la cercanía cuales estaciones deberían estar activas o inactivas. Existen estaciones entrelazadas que requieren las unas de las

otras, por lo cual se deberán tener en cuenta reglas y excepciones especiales al momento de definir las activaciones. De la misma manera, los conductores particulares también se ven en necesidad de implementar reglas específicas cuando sus componentes difieren a los de sus estaciones hermanas. La estructura general del manejador de estaciones se muestra a continuación.

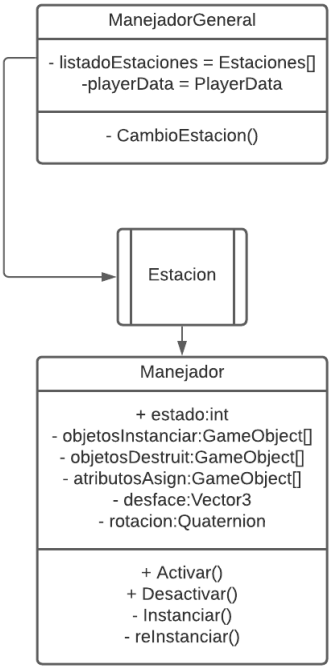


Figura 2.5.3.1: Diagrama de manejador

2.6 Prototipos

2.6.1 Manejador de estaciones

El manejador de estaciones tiene como objetivo principal instanciar elementos que interactúan con el jugador y destruirlos cuando ya no estén en uso para que el recolector de basura libere memoria. Para prototipar el manejador de estaciones es necesario considerar el formato en el cual los objetos serán almacenados, la información que pueden contener y que sus referencias a objetos con los que interactúen se mantengan intactas. Es por ello que la manera más eficiente y eficaz de prototipar es con una implementación rápida en la primera estación, siendo esta la

menos compleja, y fijando el evento de teclas en desuso (ej. tecla: y, x del teclado) como triggers que activen los eventos de instanciación y destrucción. Si la estación 1 puede ser completada sin inconvenientes, el videojuego se puede culminar y el rendimiento del videojuego no se ve afectado de forma negativa se dará paso a la implementación del módulo.

2.6.2 Ajuste de calidad gráfica

El prototipo de ajuste de calidad del videojuego tiene como premisa reducir la cantidad de objetos renderizados y nivel de detalle del entorno según la distancia entre los objetos y el jugador para disminuir la demanda gráfica del videojuego al dispositivo que lo ejecuta. Ya que esta implementación impacta de forma visual y puede ser ajustada en el transcurso de la jugabilidad, las pruebas a realizar estarán enfocadas a si las decoraciones estéticas conservan un mínimo aceptable de resolución y si se tiene permiso de acceso en tiempo de ejecución. Las mejoras en rendimiento son prácticamente seguras debido a la gran cantidad de vegetación en constante movimiento que compone el camino del sendero..

2.6.3 Mesh combiner

Prototipar el combinador de Meshes o modelos 3D pretende reducir el número de llamadas al sistema realizadas para dibujar los objetos en la pantalla del jugador, los objetos combinables son aquellos que comparten la misma textura (ej. rocas del mismo color) y la pertenencia a la misma estación. Ya que esta implementación solo impacta de forma visual, las pruebas a realizar estarán relacionadas a si los modelos 3D conservan un mínimo aceptable de resolución y si las mejoras en rendimiento son significativas.

CAPÍTULO 3

3 Resultados y Análisis

En el presente capítulo se documenta el proceso seguido para alcanzar las propuestas establecidas previamente. Se ofrecen más detalles sobre la herramienta utilizada para la medición de las métricas (Rendering, Scripting, Physic, Animation y UI), junto con las implementaciones realizadas entre las cuales están: Manejador de Estaciones, Manejador de Gráfico, Sistema de logros
Además las validaciones de las mejoras percibibles.

3.1 Recopilación de incidencias

El primer paso en el proceso de optimización es tener una base sólida del estado actual del software. Para ello se recopilaron y analizaron las incidencias de bajo rendimiento en el videojuego Mi Bosque 3D realizando el barrido en el mismo mientras se ejecutaba el profiler de Unity, el cual recogió en cada instante de tiempo las métricas de rendimiento del videojuego y las mostró en forma de gráfica. Por convención estándar en el área de los videojuegos se consideró para este proyecto los FPS como medida de rendimiento. De las variables analizadas en el profiler se optó por descartar el VSync ya que altera la medición de FPS netos mediante una amortiguación vertical y se consideró necesario limitar la muestra de datos recogidos a los momentos en que el jugador está en movimiento debido a que la variable UI/Rendering tiene una influencia.

A partir de la información recopilada se determinó que el mayor número y la mayor magnitud de las incidencias de bajo rendimiento se concentraban en la escena principal del juego, donde el jugador explora por la mayor parte de la experiencia. El origen de las incidencias más constantes fue el consumo por renderización de elementos visuales y el origen de las incidencias más críticas fue el cómputo lógico originado cada vez que se completa una misión. De manera paralela se detectó una fuga de memoria/consumo de CPU provocada por elementos activos en segundo plano

sin recibir el tratamiento debido. El listado de las incidencias se ve reflejado en el **[Anexo 1]** al final del presente documento.

3.2 Desarrollo de manejador de componentes

Como primera propuesta para mejorar el rendimiento general de la escena del bosque, el manejador de componentes fue diseñado para reducir la carga de memoria y procesamiento en todo momento mediante la instanciación y destrucción de componentes a tiempo real. De está manera la existencia de los elementos cargados se limitará a su necesidad. Los elementos indispensables para la jugabilidad que no han de ser removidos, para todos los demás existirán estados de actividad, inactividad o inexistencia según se requiera para el progreso del jugador. En la presente sección se detalla el procedimiento realizado para implementar el módulo de manejo de componentes.

3.2.1 Preparativos

El primer paso realizado fue verificar que todos los objetos dentro de la jerarquía de la escena estén siendo utilizados. Objetos completamente indivisibles para el jugador, eventos inalcanzables y cualquier prototipo de implementación no finalizada fue removido de la escena. Los elementos detectados y removidos fueron:

- Objetos pertenecientes a versiones anteriores del sendero como objetos recogibles.
- Objetos decorativos en sectores a los que el jugador no tiene permitido llegar o divisar como piedras y terrenos.
- Objetos duplicados como el sistema de iluminación del sendero o el lago
- Objetos con referencias rotas sin funcionalidad como árboles vacíos o reproductores de multimedia.

A continuación se realizó la organización de objetos según su ubicación y su tipo. Gracias a la estructura de Unity se pueden manejar jerarquías de objetos,

permitiendo su agrupación. Cada elemento fue introducido a su contenedor correspondiente como lo muestra la **Figura 2.5.2.1**. Con los elementos organizados, es posible clasificar cuáles de los elementos son necesarios en todo momento, cuales son prescindibles una vez cumplida su función y cuales podrían variar según sea el caso.

Finalmente, teniendo una clasificación ordenada de los elementos, es posible utilizar los beneficios de los Prefabs de Unity. Para el caso de las especies de flora y fauna, los límites físicos de las estaciones y los elementos decorativos que poseen mayor independencia modular es posible guardar las agrupaciones como Prefabs y eliminarlos de la escena principal para reducir la carga inicial y general de la escena con instanciaciones en tiempo de ejecución.

3.2.2 Implementación

A partir de este punto se denominará como “Listado de Prefabs” al conjunto de especies, límites físicos y la decoración correspondiente a cada estación, y “Listado de desafíos” al conjunto de elementos pertenecientes a desafíos, misiones y reproducciones de videos.

Se diseñaron manejadores específicos que por estación sean capaces de administrar los componentes de su estación respectiva y un manejador general que informe a los manejadores específicos sobre la cercanía del jugador. Los manejadores específicos poseen referencias a los Prefabs en un “Listado de Prefabs” y a los elementos de la escena en un “Listado de desafíos”.

En todo momento el jugador tendrá un rango de visión mínimo de 3 estaciones, normalmente serán su estación actual, la previa y la próxima. Mientras una estación esté dentro del rango de visión, siempre tendrá a su disposición los elementos del “Listado de Prefabs” activos, caso contrario serán destruidos. Al cargar el bosque y al progresar a la siguiente estación, cualquier desafío previamente completado es completamente destruido.

De esta manera el videojuego continuamente libera memoria mientras el jugador progresa, permitiendo que el recolector de basura realice su trabajo y reduciendo la

carga en el dispositivo sobre todo en la segunda mitad del recorrido donde presentaba mayor falencia.

3.2.3 Análisis de resultados

A continuación se muestra una tabla que compara el cambio en el rendimiento percibido gracias a las implementaciones realizadas. En todas las estaciones el cambio resulta favorable y de manera general se puede percibir una mejora aproximada del 30% en los FPS del juego.

Número de Estación	Antes (Media en fps)	Después (Media en fps)
E 1	60	75
E 2	50	75
E 3	45	80
E 4	60	75
E 5	75	90
E 6	80	100
E 7	75	95
Media	63	84

Tabla 3.2.3.1: Comparativa de mejora del manejador de componentes

De manera general, los presentes cambios se justifican en un rebalanceo de la carga del procesamiento. En lugar de tener un consumo alto y continuo como los muestra la **Figura 3.2.3.1,** se concentra la carga de instanciación/destrucción al momento de ingresar a una estación específica. Es importante mantener un juicio crítico al momento de balancear la carga entre instalaciones y permanencia de objetos debido a que una carga masiva instantánea de elementos tampoco es recomendable.

Las imágenes a continuación, extraídas de la quinta estación del videojuego Mi bosque 3D, comparan el consumo de recursos antes y después de los cambios realizados tomando como referencia los componentes del juego representados por distintas tonalidades. Las reducciones de consumo se distribuyeron a través del Rendering (verde), Physics (naranja), Scripting (celeste) y UI (morado).

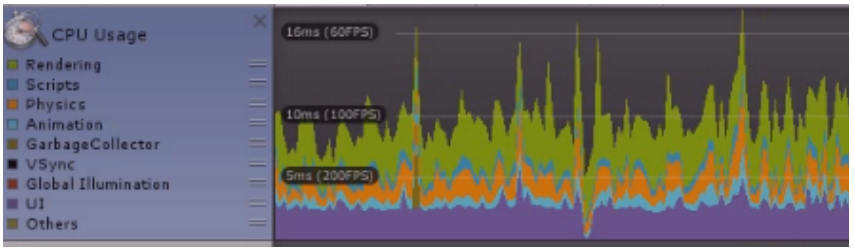


Figura 3.2.3.1: Gráfica de rendimiento previo a la implementación

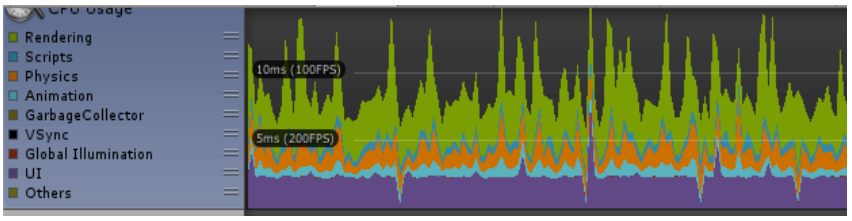


Figura 3.2.3.2: Gráfica de rendimiento posterior a la implementación

3.3 Desarrollo de manejador gráfico

Como segunda propuesta para mejorar el rendimiento general de la escena del bosque, la implementación del manejador gráfico tomó en consideración que factores afectan al consumo computacional en el componente visual del videojuego Mi Bosque 3D. Su objetivo es ajustar en tiempo real la cantidad de drawcalls que utiliza Unity para dibujar y renderizar el ambiente modificando el terreno y los elementos estéticos que lo componen.

3.3.1 Implementación

Se implementó un módulo con acceso al sistema de aves, al sistema de viento y al terreno de la escena del bosque con los permisos necesarios para editar sus atributos, componentes que mejoran la estética del ambiente sin alterar la jugabilidad.

Posteriormente se establecieron tres niveles de ajustes predeterminados para que el usuario pueda desde el menú de pausa modificar la demanda gráfica según su preferencia.

Figura 3.3.1.1: Menú de calidad gráfica

Los detalles más relevantes para ajustar la calidad del videojuego fueron las siguientes:

- **drawInstaced**: Para instanciar las drawcalls para aumentar la eficiencia de las llamadas.
- **detailObjectDensity**: Densidad de árboles del bosque que existen en el videojuego.
- **detailObjectDistance**: Distancia de visualización de detalles de los árboles.
- **shadowCastingMode**: Modo de casteo de las sombras del bosque.
- **Sistema de animación de pájaros**: GameObject encargado de animar el vuelo de los pájaros.
- **WindZone:** Zona de viento para la animación del movimiento de hojas y árboles.

A partir de las características mencionadas previamente, se establecieron estándares de modificación para sus valores en tiempo real que logren aumentar o disminuir la cantidad de drawcalls al sistema. Estos estándares se dividieron en 3 niveles:

- **Nivel bajo:** Configuración con menor demanda gráfica para el computador, en la cual se disminuye la cantidad de detalle, distancia y su modo de instanciación con respecto a las llamadas ahora es instanciada. Además las sombras son desactivadas al igual que las animaciones de los pájaros y las zonas de viento.
- **Nivel medio:** Es la experiencia estándar del videojuego para el jugador, en la cual se balancea los recursos gráficos con una distancia promedio con respecto al jugador. En esta configuración se activan las zonas de viento, la animación de los pájaros y las sombras se proyectan en modo TwoSided. Añadiendo también que se deshabilita la opción de drawCalls de forma instanciada.
- **Nivel alto:** Se aumenta la calidad gráfica para renderizar más elementos en el terreno, para mejorar la visualización del entorno. Esta es la configuración más demandante incrementando bastante el nivel de detalle del entorno, específicamente detailObjectDensity y detailObjectDistance, junto con la distancia de error para dibujar los recursos gráficos. También se incrementa la distancia de renderización de los objetos haciendo que se dibujen antes de que se pueda visualizar.

3.3.3 Análisis de resultados

A continuación se muestra una tabla que compara el cambio en el rendimiento percibido en los distintos niveles de calidad gráfica. Como referencia al nivel medio de consumo se tienen los valores posteriores a la implementación del manejador de componentes en la **sesión 3.2** del presente documento. En el nivel bajo se puede percibir una mejora general del 15% de los fps mientras que el nivel más alto posee niveles similares pero mejores a los valores obtenidos como estado inicial del software.

Número de Estación	Nivel bajo	Nivel medio	Nivel alto
E 1	85	75	60

E 2	85	75	60
E 3	90	80	55
E 4	100	75	60
E 5	100	90	80
E 6	115	100	70
E 7	105	95	75
Media	97	84	66

Tabla 3.3.3.1: Comparativa de mejora por ajustes gráficos

3.4 Ajustes realizados al sistema de logros

Como tercera propuesta para mejorar el rendimiento general de la escena del bosque, se solicitó revisar el sistema de misiones y logros. En el mismo se detectó una incidencia de rendimiento constante y recurrente detectada en el sistema de logros. Al completar misiones, el videojuego entero se detiene durante aproximadamente 6 segundos, imposibilitando al jugador actuar de cualquier manera. En la presente sección se detalla la causa y solución de la incidencia detectada.

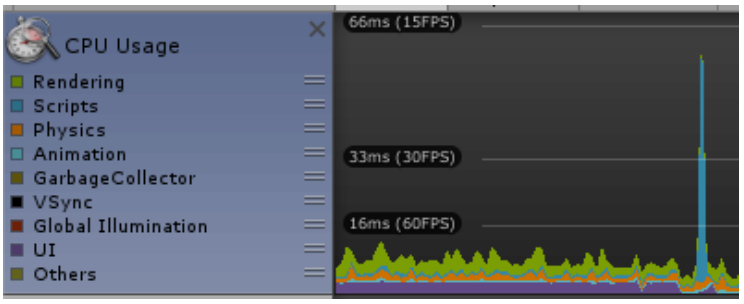


Figura 3.4.1: Pico de bajo rendimiento al adquirir logro

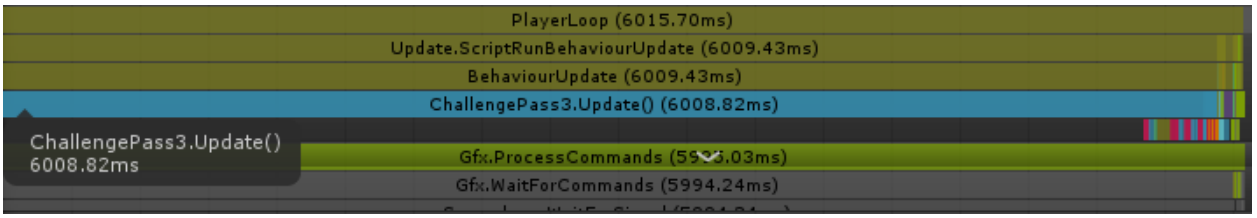


Figura 3.4.1: Duración de pico de bajo rendimiento

3.4.1 Implementación

Después de realizar un análisis exhaustivo se logró determinar que de manera independiente el sistema de misiones y logros funciona de manera adecuada. Sin embargo, los sistemas de peticiones web y el sistema de recopilación y análisis estadístico que evalúa las interacciones del usuario con el videojuego no estaban adecuadamente integrados. Esto causaba que cada vez que el jugador desbloquea un logro, el videojuego genera las estadísticas de la estación respectiva y las envía a los servidores de la ESPOL. Dichos servidores son accesibles únicamente a través de la red local para usuarios que se encuentren físicamente en la institución educativa o a través de una VPN con credenciales de un miembro de la institución.

La falla en la integración causaba una demora neta de aproximadamente 6 segundos mientras se generaban las estadísticas, se enviaban y se detectaba el timeout de la conexión. Ya que el videojuego está diseñado para poder ser jugado independientemente de la localidad o dispositivo en el que se ejecute, se implementó una interfaz que conecta el sistema de misiones con las peticiones web sin interrupciones en casos de conexiones fallidas, validando la disponibilidad de conexión con la red.

3.4.2 Análisis de resultados

Con el cambio realizado se logró normalizar el pico de bajo rendimiento al nivel de la media que los picos de las estaciones correspondientes poseen. El acto de culminar una misión se vuelve indistinguible a los procesos naturales del juego.

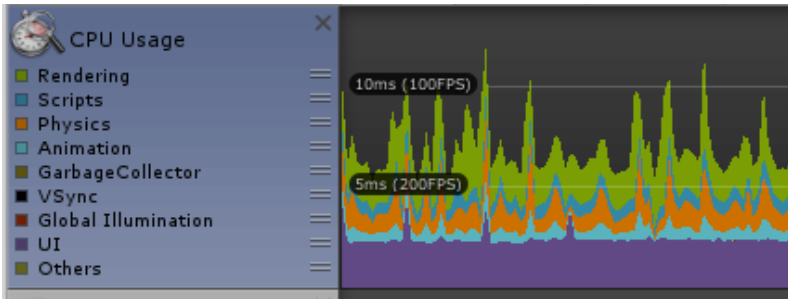


Figura 3.4.2.1: Amortiguación de pico de bajo rendimiento

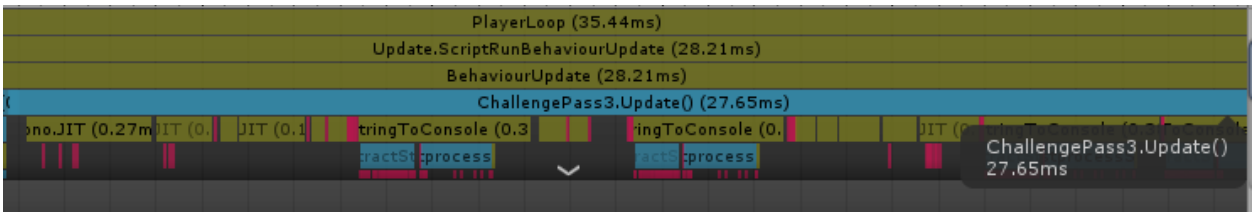


Figura 3.4.2.2: Duración de pico de bajo rendimiento amortiguado

3.5 Limpieza de recursos no utilizados

Se encontraron librerías y módulos no utilizados, los cuales ocupaban procesos innecesarios en el tiempo de juego del videojuego, restando recursos a procesos importantes.

Entre los módulos encontrados se identificaron los siguientes:

- Vehicles: Este módulo es usado para simular físicas de vehículos, pero mientras está agregado al videojuego, no aporta funcionalidades necesarias debido a que el videojuego no involucra vehículos por el momento.
- Purchasing: Este módulo es principalmente para implementar compras en el videojuego sin embargo debido a la naturaleza del videojuego no se requiere compras integradas.
- Clothing: Es una librería que permite la implementación de ropa y las físicas relacionadas, pero debido a que no se interactúa con ropa consume recursos porque se simula ropa de forma predeterminada a pesar de no usarse.

3.6 Configuraciones adicionales

Para la configuración de la visibilidad de los objetos del videojuego se configuró una área de oclusión en la cual se renderizan los recursos gráficos dependiendo de la distancia de la cámara.

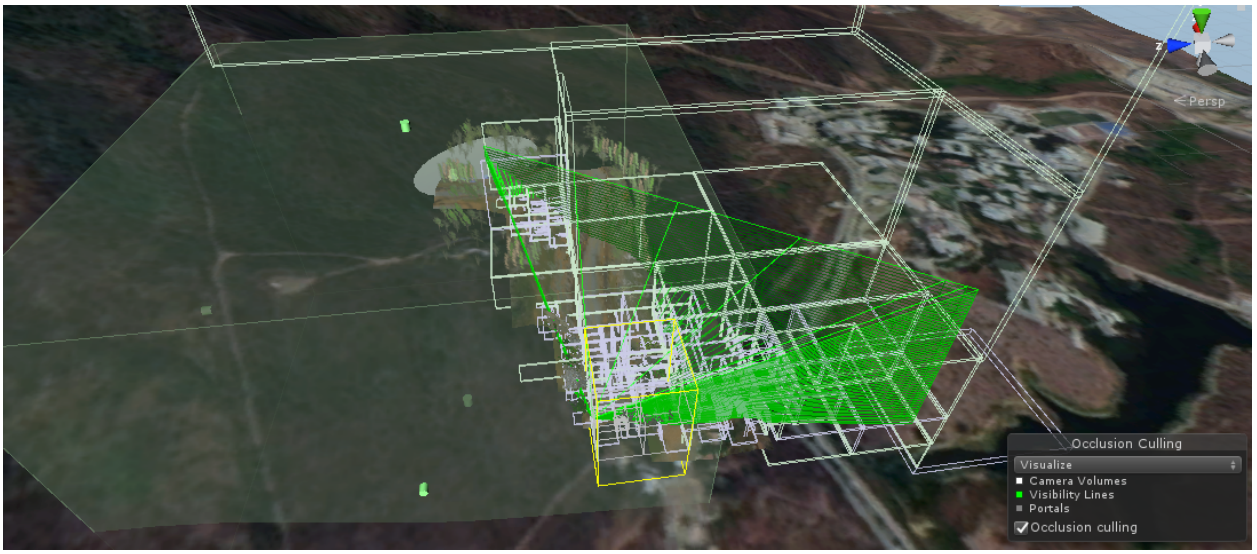


Figura 3.6.1: Área de Oclusión

Como se puede observar en la figura no toda el área está siendo calculada para ser renderizada, existen partes del mapa que no son calculadas debido a que no están dentro del rango de la cámara.

CAPÍTULO 4

1. Conclusiones y Recomendaciones

Conclusiones

- La reorganización, destrucción e instanciación en tiempo real de los recursos gráficos del videojuego mejoró la estabilidad de demanda gráfica del videojuego permitiendo una experiencia mucho más normalizada sin tantos picos de uso de los recursos del computador.
- La modificación de calidades incluyendo la animación de vuelo de los pájaros, la cantidad de follaje del bosque y las zonas de vientos permitió mayor flexibilidad a la hora de interactuar con el videojuego, debido a que permite al jugador ajustar en tiempo real la cantidad de recursos visuales habilitando la opción de abarcar mayor rango de computadoras con diferentes tipos de especificaciones técnicas.
- Las técnicas investigadas permitieron generar un documento en el cual se establecen las buenas prácticas a realizar y potenciales mejoras a futuro para soportar cambios de gran impacto sin comprometer el rendimiento del videojuego.

Recomendaciones

- Para detectar el origen de incidencias de bajo rendimiento en tiempo de ejecución lo primero que se debe realizar es un análisis de componentes que vaya de lo más general a lo más específico. Utilizando el profiler y el stats de Unity, se pueden activar y desactivar componentes de la escena en ejecución y determinar si la métrica de FPS/ms mejora de manera considerable en ausencia de dichos componentes.

- Tener llamadas sincrónicas a servicios externos, interrumpen la experiencia de videojuego drásticamente, debido a que no se procesaba paralelamente con las drawcalls y los renders, creando un cuello botella. Por lo que hay que evitarlo o volverlas asíncronas
- Limitar la renderización mostró una ligera mejora en la carga de recursos aunque tuvo un impacto ligero en comparación a las otras optimizaciones, si mejoró el control de carga en el videojuego.
- En la jerarquía de objetos de Unity es altamente recomendado encerar los valores y realizar cambios únicamente donde se los necesite (en los hijos de último nivel) para evitar futuras complicaciones con desfases de transformadas.
- Existen otros enfoques de optimización que fueron considerados y prototipos para el presente proyecto, sin embargo, debido al enfoque del videojuego y su arquitectura no presentaron resultados significativos. Para desarrolladores interesados en los otros enfoques, se recomienda investigar las siguientes implementaciones: Mesh combiner, Polycount adjuster, Arquitectura Data Driven y Singleton.
- Al trabajar con las herramientas Unity y GitHub desarrollando un videojuego de forma colaborativa, es importante mantener buenas prácticas de versionamiento y de trabajo en equipo como:
    - Mantener los commits concisos y nucleares.
    - Documentar las descripciones de manera entendible para los demás colaboradores.
    - Organizar de manera oportuna los merges de las ramas finalizadas evitando en lo posible mantenerlas activas por periodos prolongados.

BIBLIOGRAFÍA

- Paullet, Karen & Pinchot, Jamie & Rota, Daniel. (2010). Technology: Convenience or necessity. XI. 439-444.
- Li, C., Dong, Z., Untch, R., y Chasteen, M. (2013). Engaging computer science students through gamification in an online social network based collaborative learning environment. International Journal of Information and Education Technology, 3 (1), 72 – 77. doi: 10.7763/ IJIET.2013.V3.237
- Ecuadorencifras.gob.ec. 2021. Encuesta Nacional de Empleo, Desempleo y Subempleo (ENEMDU). [online] Available at: <https://www.ecuadorencifras.gob.ec/documentos/web-inec/EMPLEO/2021/Agosto-2021/5_DOCUMENTOS_ENEMDU_2021_08_DIREJ.pdf> [1 Agosto 2021].
- Unity Gaming Report 2021 (2021). Report of gaming status in the population influenced by COVID Pandemic.
- Element K. (2012). “Introduction to Software Life Cycles”
- Program Ace, (13 April, 2021). Unity vs. Unreal: What to Choose for Your Project? Comparative interest studios and businesses have shown towards building apps with Unity o Unreal development.
- Unity Documentation. Official Online Documentation for Unity, which starts from the basics to the deepest modules of Unity has to offer.
- Unity. (2020). Unity Manual. Unity 3D docs. https://docs.unity3d.com/Manual/index.html
- Unity. (2020). Unity Architecture. Unity 3D docs. https://docs.unity3d.com/Manual/unity-architecture.html
- Vargas, Ricardon Silva, Alejandra. (2019). REXE. Revista de Estudios y Experiencias en Educación. https://www.redalyc.org/journal/2431/243158860009/243158860009.pdf

APÉNDICE A

Tabla de incidencias

Evento	Tipo de Incidencia	Métrica FPS / MS	Origen	Severidad	Duración	Métrica general
Inicio de escena "MenuPartida"	Rendering	60 fps / 16 ms	Escena:MenuPartida	0	Instante	1000 fps / 1 ms
	Físicas					
	Script					
Ingreso de nombre, edad y género	Script	250 fps / 4 ms	Escena:MenuPartida	0	Instante	1000 fps / 1 ms
	Rendering					
Apertura de Select de género	Script	30 fps / 16 ms		0	Instante	1000 fps / 1 ms
Inicio escena Video	Animaciones	15 fps / 66 ms	Escena: EscenaVideo	0	Instante	1000 fps / 1 ms
			Bosque.mp4			
			Canvas>video player			
Inicio de tutorial	UI	15 fps / 66 ms	Escena: Tutorial	1	Instante*2	1000 fps / 1 ms
	físicas		FirstPersonController.cs			
			mousecontroller.cs			
	animaciones		lb_birdcontroller.cs			
			NPCcontroller.cs			
	rendering		teca, grass			
fase 1 tutorial	Script	15 fps / 66 ms	fpsController	0	Instante	100 fps / 10 ms
			BloqueoCamara			
Inicio Lobby	UI	15 fps / 66 ms	Escena: Lobby	1	Instante*2	200 fps / 5 ms
	físicas		fpsController			
Entrada a cabañas de exhibición	Script	15 fps / 66 ms	fpscontroller, FirstPersonControll	0	Instante	200 fps / 5 ms

			er.cs			
			enablemouse.cs, scrollsnaprect.cs			
	Físicas	15 fps / 66 ms	pantallaAves/pantallaAnimales			
Entrada a cabañas de proyección	Script	60 fps / 16 ms	ecuadorBio.MP4 videocontrolerlive.cs	0	Instante	200 fps / 5 ms
Inicio de escena del bosque	Script	15 fps / 66 ms	Escena: Bosque / challenges	2	media	100 fps / 10 ms
	Iluminación		light manager apagado			
	rendering		modelos de plantas, arboles, lago			
			objetos, etc			
	Físicas		FirstPersonController.cs			
Llamado a canvas Galería (Pero solo la primera vez)	UI	15 fps / 66 ms	canvas galería	1	Instante	100 fps / 10 ms
	Script		clicmouse.cs			
Rendimiento general segunda mitad del bosque						60 fps / 16 ms
Video Cadenas Tróficas	UI	30 fps / 16 ms	video cadena trofica.mp4	1	Instante	60 fps / 16 ms
	animaciones					
Completar misiones en general			SistemaLogros > LogrosGlobales.cs	3	media	60 fps / 16 ms
			notiflogros.cs ; clicmouse.cs;			
			firstpersoncontroller; challengepass			
Completar misión alimentar	animaciones	15 fps / 66 ms		3	media	60 fps / 16 ms
Completar misión reciclar	animaciones	15 fps / 66 ms		3	media	60 fps / 16 ms
Completar misión sembrar	animaciones	15 fps / 66 ms		3	media	60 fps / 16 ms
Completar misión fogata	animaciones	15 fps / 66 ms		3	media	60 fps / 16 ms

Completar misión conejo	animaciones	15 fps / 66 ms		3	media	60 fps / 16 ms

APÉNDICE B

Manual de cambios de Optimización

Manejador de Componentes

La clase encargada de gestionar la creación, destrucción y reinstantación de las estaciones es ManejadorEstaciones.cs, la cual permite instanciar, destruir e instanciar elementos según sean requeridos. Para hacer uso del manejador de componentes se necesita editar el prefab correspondiente al sector específico. Si se desea agregar una nueva especie a la “estación 1”, dicha especie debe ser agregada en “Assets -> Integradora -> PrefabsEscenas -> Estacion # -> Especies #”. De la misma manera se localizan los prefabs de la decoración y los límites de las estaciones.

Manejador de Calidad Gráfica

Para el uso del manejador de calidad se requiere usar la interfaz gráfica que se encuentra en el menú de pausa. Para la implementación del manejo de calidades se modificaron los atributos del terreno del videojuego, a continuación se detalla qué atributos se alteran para lograr ajustar la calidad gráfica.

- drawInstanced
- heightmapPixelError
- basemapDistance
- shadowCastingMode
- detailObjectDensity
- detailObjectDistance
- treeDistance
- treeBillboardDistance
- treeCrossFadeLength
- treeMaximumFullLODCount

Además de la zona de viento y el sistema de animaciones de pájaros.

Integración sistema logros / peticiones

Para solventar la espera sincrónica de las llamadas de los servicios externos se parametrizo los modos del juego para evitar invocar a los servicios cuando se esté jugando fuera de una conexión a internet contextualizada en una VPN.

Las clases que se encargan del manejo del registro del cumplimiento de las misiones son las siguientes:
ChallengePass.cs

ChallengePass3.cs
ChallengePass4.cs
ChallengePass5.cs
ChallengePass6.cs
ChallengePass7.cs

La manera en que se logra es preguntando por la propiedad OfflineMode de GameManager para recuperar el estado actual del juego. Para que en el caso de que exista conexión con internet se proceda a registrar por medio de un servicio el avance del jugador pero en caso contrario se omitirá dicha operación por motivos de rendimiento.

Área de oclusión

Se creó una área de oclusión para manejar la distancia de renderizado de los objetos dentro del videojuego. Su acceso está en la pestaña Windows -> Rendering -> Occlusion Culling.

Luces

Para disminuir la cantidad de llamadas de unity en tiempo de juego se mantuvieron las luces estáticas precalculadas para no afectar al rendimiento en general.

Uso de prefabs/static clases/scriptable

El enfoque tradicional en el área de la computación es la lectura, escritura y almacenamiento de documentos son extrayendo los datos necesarios e instanciando las clases requeridas. Sin embargo, existen 3 estructuras mayormente utilizadas para el manejo de datos en el desarrollo de videojuegos. La forma más óptima de usar estas estructuras es eligiendo de manera crítica cual trae más utilidad con menos consumo.

Los Prefabs son como el nombre lo indica, elementos preestructurados. Son capaces de funcionar con gran independencia al contexto en el cual se los instancia y albergar una gran cantidad de componentes y elementos diversos en su interior. Son extremadamente fáciles de crear. Una vez que se tiene un elemento listo y funcional en una escena, se arrastra el elemento de la jerarquía del editor hacia el explorador de archivos de Unity. Este archivo puede ser trasladado sin problema a otra escena o incluso a otro proyecto y ser usado en su nueva locación sin inconvenientes. A pesar de su facilidad de creación y uso, pueden llegar a presentar problemas de rendimiento cuando se depende únicamente de ellos. Cada instancia es única e independiente y en caso de requerirse múltiples instancias, podrían generarse redundancias de información. En el caso de Mi Bosque 3D, los prefabs podrían representar el terreno (con sus elevaciones, texturas y foliaje incluido), un grupo de rocas, pasamanos e incluso los animales y árboles (a estos últimos se les debe realizar un ajuste extra mencionados en el manejador de componentes).

La redundancia de datos al usar solo Prefabs surge cuando por ejemplo, se tienen múltiples instancias de un mismo objeto con sus atributos y métodos. Se recomienda mantener los datos comunes en una sola clase estática definida. El lado negativo de las clases estáticas es que son volátiles y cualquier cambio realizado en ellas es encerrado cuando se reinicia la escena. Además, los valores de los atributos no son visibles desde el editor de Unity.