IMPLEMENTACIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN EN LA CREACIÓN DE SOFTWARE DE VIDEOJUEGOS, BASES DE DATOS Y SERVICIOS WEB

Joaquín Badillo Granillo-A01026364, Pablo Bolio Pradilla-A01782428, Shaul Zayat Askenazi-A01783240.

"Instituto Tecnológico y de Estudios Superiores de Monterrey - Campus Santa-Fe"

Resumen

En este proyecto se presenta el desarrollo de Break Into Valhalla, un emocionante videojuego de tipo *Roguelike* RPG con apariencia 2D top-down inspirado en la mitología nórdica donde el jugador fue desterrado de Valhalla y debe demostrar su valor. Esto se logró intercomunicando una base de datos en SQL con una página web que tiene un videojuego embebido y que utiliza el protocolo HTTP para mandar y recibir datos a través de una API. La conexión con la base de datos permite a diferentes usuarios guardar en sus cuentas su progreso con nuestro juego y lo podrán visualizar en forma de gráficas.

Introducción

El videojuego fue desarrollado con el motor de Unity, mientras que el sitio web fue desarrollado con HTML, CSS y JavaScript puro. Por otra parte, la base de datos se implementó usando el DBMS MySQL, la cual se comunica con una API RESTful desarrollada en Express JS. Esta integración de tecnologías nos permite persistir datos a través de una aplicación web. Por otra parte, el código fuente está disponible en *GitHub*, tal que cualquier persona con acceso a Internet puede utilizar este proyecto como una fuente de aprendizaje. A lo largo de este informe, se detallarán los procesos de desarrollo, las tecnologías utilizadas y los resultados obtenidos con este prototipo digital.

Desarrollo

Para asegurar una experiencia única en el juego, el mapa se genera de manera procedural a partir de un árbol. Cada vértice en el árbol, representa un tipo de cuarto (cuarto de cofre, cuarto de batalla, etc.) y para 6 distintos tipos de cuarto se crearon 3 variaciones. De esta manera, el algoritmo es capaz de crear 3^6 mapas únicos; pero debido a que la posición de estos cuartos es relevante para la experiencia del jugador, los isomorfismos del árbol se pueden percibir como niveles distintos. Por ende, debido a la profundidad del árbol, se tienen en realidad $4 \cdot 3^6 = 2916$ mapas únicos en este prototipo. El uso del árbol además garantiza que cualquier usuario es capaz de encontrar objetos necesarios para su aventura como pueden ser llaves o cofres, pues si el camino existe en el grafo entonces el algoritmo garantiza que ese mismo camino existe en el nivel generado.

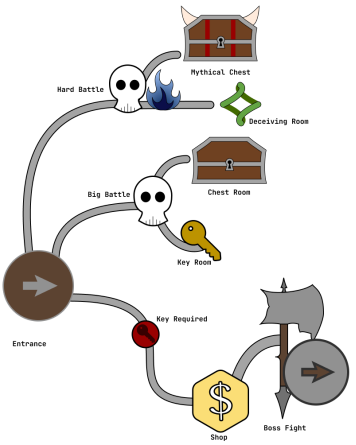

Figura 1. Abstracción del nivel

En cada uno de estos cuartos el jugador encontrará distintos tipos de enemigos. Algunos más robustos intentarán combatir cuerpo a cuerpo, alcanzando al jugador utilizando el algoritmo de A* que aproxima el mejor camino de un punto A a un punto B buscando nodos prometedores en dicho camino, al no recorrer todas las rutas posibles como lo haría el algoritmo de Dijkstra, se puede actualizar en tiempo real, por lo que se actualiza en cada cuadro sin ralentizar el juego. Por otro lado, los enemigos a distancia calculan la posición del jugador y atacan periódicamente al jugador disparando flechas; no obstante este puede redireccionarlas al atacarlas en un lapso de tiempo determinado. Esto es posible y computacionalmente eficiente, ya que en Unity las rotaciones se manejan internamente con cuaterniones, aunque para facilitar la implementación y lectura del código, las rotaciones en el código fuente se establecen con ángulos de Euler.

Además, debido a la naturaleza de un juego RPG, las decisiones que toma el usuario pueden ser el

factor determinante entre una victoria y una derrota. Esto se ve reflejado desde el primer instante en el que un jugador crea una cuenta, pues debe escoger entre los arqueros, *berserkers* y los misteriosos hechiceros. Para que esto fuera posible y gracias al uso de la programación orientada a objetos, se tuvieron que crear clases abstractas para los personajes (incluyendo a los enemigos) y para sus comportamientos de ataque.

Figura 2. *ER y RDBMS.*

La base de datos, al ser una basada en SQL, sigue el modelo entidad relación, que guarda la información en forma de tablas y mantiene relaciones a partir de *llaves foráneas*. Para reducir las libertades que pueda tener un conector a la base de datos al igual que para optimizar consultas frecuentes, se hicieron vistas, tablas virtuales que contienen la información pertinente y evitan el contacto con la base de datos como tal.

Por otro lado, existen disparadores que se activan cuando un evento ocurre; estos se utilizaron para borrar todas las tablas que tengan relación con un usuario en el momento en el que este se borre. Además, para crear partidas, cuentas de usuario nuevas y añadir registros tanto de las ubicaciones en las que murieron los usuarios como del tipo de enemigo que los eliminó, se utilizaron procedimientos. Estos permiten realizar múltiples operaciones desde una sola llamada, evitando que los usuarios de un conector a la base de datos hagan este tipo de operaciones manualmente y como las proporciona el administrador de la base de datos garantizan que se realicen adecuadamente.

Para lograr una integración fluida y eficiente entre la base de datos, el videojuego y el sitio web, se implementó una REST API (Interfaz de Programación de Aplicaciones), la cual utiliza el protocolo HTTP y se comunica por medio de JSON (Notación de objetos JavaScript).[1]

Figura 3*. Interconexión entre cliente y servidor de base de datos.*

La API funciona como un intermediario que permite hacer peticiones por medio de endpoints a un servidor que asegura la entrega de la información en un tiempo determinado. La conexión con el videojuego permite al mismo leer en la base de datos la información que será utilizada en el videojuego, así como escribirla en la misma. Por ende se tuvieron que identificar el tipo de operaciones que requería el videojuego (*insertar* datos de nuevos jugadores, actualizar métricas, leer estadísticas, etc.) para poder especificar e implementar dichos *endpoints*.

A través de los datos acumulados se pudo obtener información relevante sobre el comportamiento de los jugadores y los elementos del juego. Esto proporcionó la capacidad de mostrar información competitiva, así como ajustar las propiedades del juego para asegurar una experiencia más equilibrada y satisfactoria. Estos datos se muestran de forma gráfica en un dashboard que también le aporta transparencia a los jugadores.

Resultados

Los resultados obtenidos en este proyecto demuestran el éxito en la creación de un entorno integral que combina un videojuego embebido en una página web y una base de datos. Gracias a la integración de múltiples tecnologías, se logró proporcionar a los usuarios una experiencia fluida y sin interrupciones.

El videojuego fue incorporado dentro de la página web, lo que permitió a los usuarios acceder y disfrutar del juego directamente desde el sitio. Además, se estableció una conexión sólida y confiable con la base de datos, lo que brindó a los jugadores la posibilidad de almacenar y acceder a la información de sus partidas en cualquier momento.

Conclusiones

Nuestro proyecto, al ser de tipo Open Source[2], está alineado con el objetivo de fomentar la educación de calidad. A través de la participación de la comunidad en el proceso de desarrollo, promovemos un enfoque colaborativo que busca mejorar constantemente la calidad de la educación. La apertura del código fuente permite que diferentes actores, incluidos estudiantes, educadores y expertos en el campo, contribuyan con sus conocimientos y habilidades para enriquecer el proyecto. Esta participación activa puede llevar a mejoras significativas, correcciones de errores y la incorporación de nuevas características que se traduzcan en una experiencia de aprendizaje más enriquecedora y efectiva.

En términos de seguridad, es importante destacar que, debido a consideraciones de facilitar el despliegue del prototipo, nuestra API se encuentra actualmente en un estado público. Esta configuración conlleva ciertos riesgos, ya que la API puede ser vulnerable a inyecciones de comandos de SQL, lo que podría comprometer la integridad de la base de datos. Además, el servidor no tiene restricciones en cuanto a los dominios que pueden acceder a él, lo que aumenta el riesgo de exposición a ataques no deseados.

En cuanto a la autenticación y acceso a los recursos, actualmente no se implementa un sistema de API Tokens, lo que significa que cualquier persona puede autenticarse y acceder a los recursos disponibles. Esto puede poner en peligro la seguridad y privacidad de la información almacenada en la API.

Además, el servidor del frontend no cuenta con un sistema de sesión, lo que lo hace vulnerable a ataques de bots o en general de denegación de servicios (DoS) y puede afectar negativamente la calidad del servicio. Por otro lado, no se ha implementado un sistema de cifrado extremo a extremo, lo que implica que las contraseñas de las cuentas no están protegidas adecuadamente y al transmitir datos a través de Internet pueden sufrir ataques de MITM (Man In The Middle)[3] o bien que algún agente externo se aproveche del API público para tomar esta información. Por ende se advierte a los usuarios que no utilicen contraseñas verdaderas.

Con respecto a la optimización de la memoria caché, es posible aprovechar esta funcionalidad para almacenar consultas frecuentes y únicas. Esto permite ahorrar recursos del servidor al evitar realizar consultas directas a la base de datos, ya que la caché del dispositivo ya contiene la información almacenada. Al hacer uso de la memoria caché, se reduce la carga en el servidor y se mejora la eficiencia en el acceso a los datos, lo que se traduce en tiempos de respuesta más rápidos y una mejor experiencia del usuario.

Los permisos de acceso a la base de datos presentan una brecha en la que los usuarios pueden actualizar su información, lo que puede alterar las estadísticas y corromper el juego. Es necesario abordar esta vulnerabilidad para garantizar un control adecuado sobre las actualizaciones de los usuarios y proteger la integridad del juego.

El proyecto requería que el esquema de la base de datos cumpliera con la tercera forma normal y permitiera la escalabilidad de la normalización hasta la quinta forma normal, debido a su naturaleza relacional. Esto asegura que la base de datos esté libre de redundancias y anomalías, y permite un diseño óptimo para futuras expansiones y modificaciones. Otra opción sería pasar a un esquema no relacional (NoSQL) donde no es necesario considerar una estructura fija para las tablas facilitando la adaptación sobre los cambios que sufran los datos.

Referencias

[1] Red Hat. (Mayo 2020). *¿Qué es una API de REST?*

https://www.redhat.com/es/topics/api/what-is-a-rest-api

[2] Red Hat. (Octubre 2019).*What is open source?*

https://www.redhat.com/en/topics/open-source/what-is-open-source

[3] WeLiveSecurity. (2023, 4 abril). *Qué es un ataque de Man-in-the-Middle y cómo funciona*

https://www.welivesecurity.com/la-es/2021/12/28/que-es-ataque-man-in-the-middle-como-funciona/