Wolfenstein 3D

Manual de Proyecto

Integrantes:

Capelli Sebastián
Kristal Juan Ignacio
Pardo Lucía
Raveszani Nicole Denise

¡Bienvenidos a Wolfenstein 3D!

1. División de tareas

- Sebastián Capelli: Encargado de desarrollar la comunicación entre servidor y cliente, así como también la creación de la inteligencia artificial en Lua y el planteamiento de lógica de los Eventos.
- Juan Ignacio Kristal: Encargado del desarrollo del cliente, aplicación de raycasting e implementación de visuales y sonidos con SDL.
- Lucía Pardo: Encargada de desarrollar toda la lógica del lado del servidor, manejando los eventos y controlando cómo se desarrolla el juego.
- Nicole Denise Raveszani: Desarrolló el Editor junto a las ventanas iniciales con las que debe interactuar el usuario al comienzo de la partida.

2. Inconvenientes Encontrados

Del lado de la lógica del servidor uno de los problemas encontrados fue la escala que fue tomando el proyecto en comparación al modelo inicial planteado, y con ello el incremento en la complejidad para ir agregando funcionalidades como el Rocket Launcher.

Desde el lado del cliente el funcionamiento del algoritmo de raycasting si bien fue entendido sus simplificaciones son un tanto complejas y hace que algoritmo sea inconsistente para diferentes tipos de raycasting (entre objetos y paredes).

Por parte de las comunicaciones, en una primera instancia se opto por utilizar protocolo de texto con la intencion de reemplazarlo por protocolo binario a futuro. Debido a otro problema que dificultaba la posibilidad de copiar eventos se decidio usar la serializacion de texto del evento como informacion a guardar en las colas de eventos tanto del server como del cliente. Lo cual generó un acomplamiento tal que no se podía reemplazar la cadena de strings por una cadena de bytes sin tocar la lógica del juego. De todas formas, se hizo uso de las funciones ntohs y htons para incluir en la tira de bytes que se envian y se reciben entre los sockets por medio del objeto Protocol.

Tanto en el editor como en la interfaz del cliente, QT presentó algunos inconvenientes. Cada vez que se agregó un nuevo .ui, se debió realizar una nueva build, debido a que realizar únicamente la compilación marcaba errores de QT. Correr Valgrind sobre una aplicación que utiliza QT muestra leaks correspondientes a la librería y no al código propio, por lo que se tuvo que utilizar un archivo para suprimir los leaks de las librerías ajenas. Varios de los widgets de QT toman el ownership de los objetos que se les agregan, por lo que al código escrito pareciera faltarle los correspondientes delete de los objetos instanciados con new. Se documentó en los destructores de las clases que utilizan estos objetos de QT la razón por la cual no se realizan los deletes. QT no ofrece mucha libertad cuando se utiliza css en el estilo de los widgets, ya que hay algunas opciones de css que QT no soporta. Un ejemplo de esto se puede ver en la pantalla donde se selecciona la partida al que el usuario desea unirse, el header de la lista de partidas puede parecer blanco o gris oscuro según la computadora del usuario. Esta parte de la pantalla no se pudo personalizar. Por último, tras la reacomodación de carpetas, no pudimos volver a agregar un .ui a la interfaz de cliente, en específico, ScoreScreen.ui, ya que no generaba el ui_ScoreScreen.h al momento de compilar mientras que los demás archivos del resto de los .ui eran generados. Finalmente, decidimos compilar a parte el archivo para que se genere el ui_ScoreScreen.h que se pueda compilar la aplicación. El único inconveniente de esto es que,

si se desea modificar el .ui en una aplicación de QT se debe volver a compilar a parte para actualizar el ui_ScoreScreen.h.

3. Herramientas

Algunas herramientas auxiliares utilizadas a lo largo del trabajo práctico fue el IDE Clion, perteneciente a JetBrains, para el uso de refactors así también como debbuging en conjunto a gdb. Valgrind fue otra herramienta utilizada para lograr encontrar pérdidas de memoria. También se utilizó librerías de QT y la aplicación QT Designer para la interfaz de usuario del juego y del editor de mapas.

El proyecto fue manejado utilizando GitHub para control de versiones.