Manual de Proyecto - Need For Speed

1. Integrantes y Organización

Padrón	Nombre y Apellido	Rol Principal / Módulos a cargo
112202	Felipe FIALAYRE	Protocolo, Concurrencia, Performance en memoria, Interfaz Qt, Sincronización Cliente/Servidor.
111829	Martino NERVI	Lógica de negocios del Servidor, Física, Parseo y lógica de archivos YAML.
111813	Francisco SERRANO	Cliente, Renderizado SDL, Interfaz gráfica Sonido, Instalador.

Metodología de Trabajo

Dado que el grupo ya se conocía y había trabajado junto anteriormente, pudimos mantener una dinámica de desarrollo fluida desde el principio. Nos organizamos mediante múltiples reuniones semanales para integrar lo que hacía cada uno y planificar coordinadamente las siguientes tareas. Optamos por avanzar en bloque, desarrollando funcionalidades completas (*end-to-end*) antes de pasar a la siguiente, lo que nos permitió asegurar que el proyecto se mantuviera estable en cada etapa sin dejar cabos sueltos.

Cronograma Real:

- **Semana 1-2:** Setup del proyecto, Skeleton del Threading/Concurrencia y Sockets básico.
- **Semana 3-4:** Terminando concurrencia (múltiples partidas), lógica de choques, inicialización de autos, movimiento (del lado del servidor) con todos los mensajes que eso significa. Interfaz gráfica renderiza correctamente tanto el mapa en background como los autos en su posición “real” en el mundo de Box2D. Desarrollo completo del mapa en Box2D. Minimapa disponible.
- **Semana 5-6:** Lógica de lobby en QT arreglando problemas de concurrencia, estados inválidos, unirse y salir de partidas (estado del servidor sincronizado con interfaz gráfica a tiempo real). Lógica de estadísticas post-partida. Fix del flujo del programa para cerrar y abrir las pestañas de forma coherente con lo que haga el cliente. Agregando lógica de compras (del lado del servidor y cliente) con integración en la interfaz gráfica. Sonido en

SDL y pulido casi total desde el punto de vista gráfico. El Minimapa renderiza recorridos recibidos del servidor.
- **Semana 7:** Integración de un flujo del programa limpio, múltiples carreras por partida, lógica de compras en cada carrera, implementación de lógica de cheats in-game, pantalla de compras mejorada. Cierre ordenado de hilos sin leaks de memoria del lado del servidor y cliente. Ajustes finales, tests de protocolo, corrección de bugs y documentación.

2. Herramientas Utilizadas

Entorno de Desarrollo

- **IDE/Editores:** Visual Studio Code, CLion.
- **Control de Versiones:** Git + GitHub (Branching model: Feature branches).
- **Diagramación:** PlantUML para diagramas de secuencia y clases.

Análisis y Calidad de Código

- **Valgrind:** Se utilizó para detectar memory leaks. El informe final muestra 0 bytes perdidos en ejecución.
- **LLDB/GDB:** Para debugging de segmentation faults durante el desarrollo.
- **Clang-format:** Para mantener el estilo de código.

3. Retrospectiva y Conclusiones

Puntos Problemáticos (Desafíos)

1. **Sincronización de Hilos:** Por más que la idea del flujo no fue tan difícil de imaginar fue difícil mantener el estado del servidor sincronizado con los clientes para evitar accesos inválidos a recursos liberados y hacer dicho flujo robusto para que independientemente de que haga el cliente nunca se caiga nada ni hayan mensajes inesperados o no manejados.
2. **Integración Física-Vista:** Lograr que la interpolación en el cliente se viera fluida a pesar de la latencia de la red.
3. **Librerías Gráficas:** Aprender el ciclo de vida de Qt para integrarlo o separarlo del loop de SDL fue un desafío de diseño.

¿Qué cambiaríamos si empezáramos de nuevo?

- Definir el protocolo de comunicación de forma más estricta desde el día 1.
- Usar una librería de serialización más robusta en lugar de armar los bytes manualmente al principio.
- Implementar tests unitarios automatizados para la lógica del juego desde el principio del desarrollo para hacer más simple el desarrollo.