🗒️ Manual de Proyecto – Counter Strike 2D

👥 Integrantes y Tareas

Sebastián Kraglievich	Desarrollo del cliente y la interfaz gráfica utilizando SDL2
Agustin Perez Romano	Implementación del protocolo de comunicación y manejo de hilos en el servidor
Morena Sandroni	Desarrollo del editor de niveles y menús con Qt
Mateo Bulnes	Desarrollo de la lógica del juego en el servidor

---

📆 Organización por Semana

Durante las primeras semanas nos dividimos las tareas para que cada integrante pudiera avanzar de forma paralela. Establecimos reuniones semanales de integración donde mostramos avances y conectamos las partes desarrolladas individualmente.

En las últimas semanas, aumentamos la frecuencia de las reuniones para poder unir correctamente todas las piezas del proyecto (cliente, servidor, editor, comunicación, lógica) y resolver bugs de integración.

🛠️ Herramientas Utilizadas

- **Lenguaje:** C++20
- **Entorno de desarrollo:** CLion
- **Compilación:** CMake
- **Librerías:**
    - SDL2 para gráficos (cliente)
    - SDL2pp como wrapper de SDL
    - Qt6 para el editor de niveles
    - yaml-cpp para parseo de archivos

- **Control de versiones:** Git + GitHub
- **Testing:** Google Test
- **Documentación y diagramas:** PlantUML, Google Docs

📚 Documentación y Recursos Consultados

- Documentación oficial de SDL2
- Repositorio libSDL2pp
- Tutorial de libSDL2pp
- Documentación oficial de Qt6

❗ Dificultades Encontradas

- La adaptación al pensamiento orientado a juegos y en particular al **patrón component** fue uno de los mayores retos.
- Asegurar que la sincronización entre cliente y servidor funcione en tiempo real sin desfasajes visuales fue especialmente desafiante.

✅ ¿Pudimos llegar con todo?

✔️ Sí, logramos entregar un juego completamente funcional con todas las funcionalidades requeridas: partidas multicliente, comunicación en red, ronda con sistema de fases, bomba, armas, colisiones, y un editor de mapas funcional.

🔁 ¿Qué cambiaríamos si lo hiciéramos de nuevo?

A nivel código:

- Modularizar desde el inicio pensando en integración futura.
- En la lógica del servidor en Match, se podría haber implementado una abstracción de un procesador de acciones, para evitar tener el switch tan extenso en Match::processAction

- Se podría haber implementado un RoundHandler que maneje las transiciones entre rondas, y que toda esa lógica no esté directamente en Match

A nivel organizacional:

- Acelerar el desarrollo de menús y pantallas previas a la partida en las primeras semanas.

- Dejar más tiempo al final exclusivamente para integración y testing profundo.

- Documentar más explícitamente las interfaces entre los módulos desde el principio.
