TP - Final Argentum

TP final taller de programacion 1

URL: https://github.com/FrancoMariotti/tp-final-argentum

Manual de Proyecto

1. Integrantes:
    i. Alasino Franco 102165
    ii. Mariotti Franco 102223
    iii. Leguizamon Agustin 99535

2. Enunciado:

Juego: Argentum Online (AO)

El objetivo de este trabajo práctico es hacer una versión simplificada del juego argentino MMORPG Argentum Online. Simplificado dado que la cantidad de mapas, items y monstruos es mucho menor a la del juego original.

El juego trata de que cada jugador conforma parte de un mundo habitado por NPCs (criaturas controladas por el juego) y otros jugadores. Los jugadores pueden combatir contra estas criaturas y hasta otros jugadores para obtener oro, items y subir de nivel para volverse cada vez más fuerte. En el juego existen ciudades donde podrán comerciar, sanar sus heridas y depositar sus objetos, además estarán protegidos de criaturas y jugadores hostiles ya que las criaturas no pueden acceder y no se puede combatir en las ciudades.

3. División de tareas:

La primer semana decidimos comenzar con la división de tareas propuesta por la cátedra: Franco Mariotti se encargaría de las tareas asignadas al alumno 1, Agustin a las del alumno 2 y Franco Alasino a las del alumno 3. Al cabo de la primer semana nos dimos cuenta que quizás las tareas asignadas al alumno 1 eran bastante demandantes para ser realizadas por un único alumno, por lo que decidimos que Franco Alasino en vez de dedicarse a las tareas del alumno 3 pasaría a trabajar junto con Franco Mariotti en el desarrollo del servidor ,modelo y lógica de juego.

Finalmente la división de tareas quedó de la siguiente forma:

Franco Mariotti	Agustin Leguizamon	Franco Alasino
Modelo y conexión	Cliente y login	Modelo y persistencia

4. Cronograma propuesto y cronograma real

Cronograma/ Semana	Propuesto		Real	
	Cliente	Servidor	Cliente	Servidor
Semana 1	-Mostrar imagen y animación. -Mostrar texto por pantalla -Reproducir sonidos, música.	-Lógica de movimiento del jugador y npcs -carga de mapas	-Mostrar imagen - Captura de eventos por teclado.	-Lógica de movimiento de personajes y npc
Semana 2	-Mostrar todo el mapa, incluyendo varios tipos de terrenos distintos, edificios y unidades (jugadores y NPC). -Mostrar la interfaz gráfica (barra de experiencia, items, vida)	-Lógica de ataque, defensa, drops tanto de jugadores como de NPCs) -Razas y clases.	-Mostrar la interfaz (consola, inventario, stats)	-Carga de mapa(obstáculos) -Lógica de ataque,defensa -Razas y clases.
Semana 3	-Mostrar los objetos (drop). -Interacción por parte del usuario. -Mini-chat tanto para leer los mensajes como para escribirlos.	-Lógica de las ciudades, -“fair play”. -Compra, venta y otros comandos. -Experiencia y niveles.	-Mostrar jugadores y NPC - Mostrar mapa - Interacción con la interfaz	-Experiencia y niveles. -Algunos comandos. -Lógica de ciudades
Semana 4	-Cliente completo. -Pantalla de login.	-Sistema de comunicación (cliente - servidor) -Servidor completo. -Persistencia y	-Mostrar drops de npcs y jugadores -Reproducir sonidos, música.	-Todos los comandos completos -drops de npcs y jugadores. -“fair play”

		configuración.		
Semana 5	-Pruebas y corrección sobre estabilidad del servidor.	-Pruebas y corrección sobre estabilidad del servidor.	-Cliente funcionando con pseudo-comunicación(proxy)	-Juego completo funcionando con pseudo-comunicación(proxy)
Semana 6	-Correcciones sobre Primera entrega -Testing y corrección de bugs -Documentación	-Correcciones sobre Primera entrega -Testing y corrección de bugs -Documentación	-Log in con Qt	-Sistema de comunicación cliente-servidor -Persistencia -Configuración
Semana 7	-Testing - Documentación - Armado del entregable	-Testing - Documentación - Armado del entregable	- Testeo y corrección de bugs - Correcciones -Armado entregable	-Testeo y corrección de bugs -Correcciones - Armado entregable

5. Inconvenientes encontrados:

- Inconveniente con serialización y los ceros:

Al enviar la capa de obstáculos,nosotros debíamos enviar un vector de enteros que contenía en su mayoría 0.Para ello ,serializabamos el mensaje lo enviabamos y el cliente lo deserializar y con esa información dibujaba los obstáculos en pantalla.Al hacer esto nos encontramos con el problema de que al deserializar este mensaje ,nos llegaba una secuencia incompleta,es decir, no podíamos deserializar toda la información del vector correctamente.Este fue el problema más grande que tuvimos en este proyecto, ya que nos tomó varios días poder encontrar la causa.Finalmente, descubrimos que el problema se encontraba al pasarle un char * a la función unpack encargada de serializar la cadena de bytes recibida por el cliente. Esta interpretaba que lo que se quería deserializar era una cadena y al encontrarse con un 00 finalizaba la serialización y generaba un resultado erroneo. La solución a este problema fue castear el puntero a un (unsigned char *), que hace que la función unpack realice una des serialización binaria y genere un resultado correcto.

- Problema al renderizar los layers del suelo y obstáculos:

Inicialmente teníamos un mapa que por clave tenía el id de la imagen a renderizar y como valor las posiciones del mapa en donde se debían renderizar.Luego una vez que cargamos ese mapa ,al iterar íbamos renderizando las imágenes en las posiciones correspondientes.El problema que teníamos era que había algunas posiciones en donde se

debían renderizar una imagen sobre otra,por ejemplo primero debíamos renderizar una imagen de pasto y sobre esa una piedra, y no se estaban renderizando en el orden correcto.El problema estaba que el mapa guardaba las claves en orden numérico, por lo tanto si la roca tenía un id = 1 y el pasto un id = 10 primero se renderizaba la piedra y luego el pasto, haciendo que solo estuviese visible el pasto.Finalmente resolvimos este problema separando las capas a renderizar en dos mensajes,primero enviamos un mensaje con la capa del suelo, esta se renderiza y luego se recibe la capa de obstáculos y se los renderiza.Asegurando así que los obstáculos se rendericen sobre las imágenes del suelo y estén visibles al cliente.

6. Analisis de puntos pendientes:

- Mapa: Decidimos hacer un mapa pequeño y no le dedicamos tanto tiempo a su diseño, si se quisiera se podría diseñar un nuevo mapa en tiled ,exportarlo como Json ,poner la nueva información en el archivo de configuración del juego ,agregar las imagenes al cliente ,y luego el juego podría cargarlo correctamente
- Efectos especiales: A pesar de tener los efectos de los hechizos con las pistas de audio implementados, no llegamos a agregar el mensaje y la lógica por parte del servidor para agregar los efectos (explosion, misil, curar, etc) a los personajes y NPCs del juego.
- Sonidos: Faltan pistas de audio para golpes con armas, y sonidos de los NPC.

7. Herramientas:
    i. Tiled: Para la creación del mapa y conversión de su información a formato Json.
    ii. Clion: IDE utilizado
    iii. PlantUml: Para la creación de diagramas UML
    iv. QtCreator: Para diseño de la interfaz
    v. Wireshark: Para el analisis de los paquetes enviados.
    vi. Git: Para el control de versiones
8. Conclusiones:

Desde el grupo creemos que este proyecto fue útil para acercarnos al mundo laboral, es decir, darnos una idea de como se desarrolla un proyecto informático en la “vida real”.Desde como desarrollar en grupo,dividiendo tareas e integrando los avances que cada integrante iba haciendo ,hasta como documentar un proyecto, que información debe acompañar el código, estandares de codificacion,entre otras cosas. También fue una oportunidad para descubrir nuevas herramientas que facilitan el desarrollo de un proyecto: características más avanzadas de Clion, utilizar Json para crear un archivo de configuración, bibliotecas gráficas y de serialización.